Institute for Space and Nuclear Power Studies
School of Engineering
The University of New Mexico
Albuquerque, New Mexico 87131–1341

USER'S MANUAL for "HPTAM", A TWO-DIMENSIONAL HEAT PIPE TRANSIENT ANALYSIS MODEL, INCLUDING THE STARTUP FROM A FROZEN STATE

JEAN-MICHEL TOURNIER AND MOHAMED S. EL-GENK

FINAL REPORT NO. UNM-ISNPS-4-1995

NASA Grant No. NAG3-941
to
Institute for Space and Nuclear Power Studies
University of New Mexico
Farris Engineering Center, Rm. 239
Albuquerque, NM 87131

September 1995

ABSTRACT

This report describes the user's manual for "HPTAM," a two–dimensional Heat Pipe Transient Analysis Model. HPTAM is described in detail in the UNM–ISNPS–3–1995 report which accompanies the present manual. The model offers a menu that lists a number of working fluids and wall and wick materials from which the user can choose. HPTAM is capable of simulating the startup of heat pipes from either a fully–thawed or frozen condition of the working fluid in the wick structure. The manual includes instructions for installing and running HPTAM on either a UNIX, MS–DOS or VMS operating system. Samples for input and output files are also provided to help the user with the code.

Acknowledgments

This research has been funded by NASA Lewis Research Center under Grant N°. NAG3–941 for a number of years. When funding was discontinued in March 1994, work was completed with internal funds by the Institute for Space and Nuclear Power Studies (ISNPS). The authors are grateful to NASA Lewis for the financial support to undertake this challenging and demanding research. We wish to thank Mr. Albert Juhasz, the NASA Lewis Technical Officer of the Grant, for his continuous support and constructive suggestions and input throughout the development of HPTAM. We are also grateful to NASA Lewis Management, Mr. Joseph Sovie and Dr. Henry Brandhorst for their encouragement and support. Finally, we wish to acknowledge the help of the ISNPS staff, and the financial support provided by ISNPS. Without it, completing the development and documentation of HPTAM would not have been possible.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

1. INTRODUCTION

HPTAM, a two-dimensional Heat Pipe Transient Analysis Model, was developed at the University of New Mexico's Institute for Space and Nuclear Power Studies to investigate and simulate the transient operation of heat pipes for space applications, including the startup from a frozen state. Heat pipes are highly reliable and efficient energy transport devices which have been considered for many terrestrial and space thermal management applications. Heat pipes can transport large amounts of energy over a long distance with a small temperature drop by means of liquid evaporation at the heat source (evaporator section) and vapor condensation at the heat sink (condenser section). Capillary force developed in a wick structure returns the liquid back to the evaporator region. The model developed at UNM can handle both cylindrical (circular heat pipe) and rectangular (slab heat pipe) geometries, and divides the heat pipe into three regions, wall, wick and vapor regions. The wick structure can be a homogeneous porous medium, a wire-screened wick, or an open annulus. The model solves the enthalpy, mass, and momentum conservation equations in both the wick and vapor regions. HPTAM can incorporate any acceleration effect in the axial direction, and a radial centripetal force for simulating rotating heat pipes. In this version of the code, the thermophysical properties of lithium, potassium, sodium, and water working fluids have been incorporated, as well as that of several structural materials (carbon, tungsten, niobium, zirconium, stainless-steel, and copper). More details on the physical model in HPTAM can be found in the adjoining report entitled "HPTAM, a Two-Dimensional Heat Pipe Transient Analysis Model, Including the Startup from a Frozen State," and in the adjoining publications.

The code is written in Standard Fortran 77 and is easily implemented on any machine supporting a Fortran 77 compiler. In the present effort, HPTAM was successfully implemented on IBM-PC compatible machines (386 and 486 running MS-DOS), on machines running VMS operating system (DEC VAX 6320), and the UNIX operating system (Sun workstations, AIX-RS6000 IBM machines, CRAY-YMP).

Included in this package is a 3.5" MS-DOS formatted disk containing the four Fortran source files of the code, along with a standard input file and three sets of command files, one for each operating system (MS-DOS, UNIX, and VMS). The code automatically generates a set of graphs showing temporal and spatial variations of various quantities of interest, such as pressures, temperatures, velocities, etc... These outputs can be generated in a form compatible with the TELLAGRAF or EASY–PLOT software packages, whether a VMS operating system machine or an IBM-compatible Personal Computer (running MS-DOS) is available to the user. In case the code is running on a UNIX machine, it is necessary that a VMS machine or a MS-DOS machine be used to visualize the plots generated by the code. For example, the File Transfer Program (FTP) can be used to transfer the output files generated on the UNIX machine to the MS-DOS machine via the high-speed Ethernet network, in order to visualize these plots with the EASY–PLOT software.

In the following section of this manual (Section 2), a quick description of the Fortran source files will be given, and the major subroutines of the program will be outlined. Section 3 will focus on the input file and its parameters. The following sections of the manual will describe the input and output files of HPTAM (Section 4) and how to install and run the code on UNIX (Section 5), MS-DOS (Section 6), and VMS machines (Section 7). The following notations will be used throughout all the sections: the prompts will be represented by the characters ">," "%," and "$" on MS-DOS, UNIX, and VMS machines, respectively. All prompt commands will be typed normally; filenames, variable, and subroutine names will be typed in **bold**. Certain filenames and directory names are left to the discretion of the user; in such case, the proposed name in the manual will by typed in ***bold and italic.***

2. THE FORTRAN SOURCE FILES OF HPTAM

The physical model in HPTAM is thoroughly described in the adjoining report and publications. The vapor flow in the heat pipe is modeled using the compressible Navier-Stokes equations. The liquid flow in the wick is calculated using the Brinkman-Forchheimer-extended Darcy's continuity and momentum equations. The phase-change in the wick is modeled using the volume-averaged homogeneous enthalpy method. The liquid and vapor phases are thermally and hydrodynamically coupled using the mass, momentum and enthalpy interfacial jump conditions. The radial momentum jump condition is the capillary relationship of Pascal, and the axial momentum condition is replaced with a no-slip flow condition to model the continuum flow regime. The evaporation, condensation, sublimation, and resolidification interfacial mass rates are calculated from the kinetic theory of gases, and HPTAM implicitly predicts the radius of curvature of the liquid meniscus along the heat pipe. The Free-molecule and transition Flow regimes are modled using a one-dimensional vapor model based on the Dusty Gas Model. The governing equations and boundary conditions are discretized on a staggered grid using the control-volume integration approach proposed by Patankar, and solved using a finite-difference discretization method, based on a SIMPLEC segregated iterative solution technique developed by Tournier and El-Genk.

The code is formed of four Standard Fortran 77 source files, namely **hptam.for**, **subnew.for**, **subold.for**, and **poisson.for**. The file **hptam.for** contains the main program unit, as well as the iterative Strongly Implicit Solver for the solution of the enthalpy and momentum linear systems. In the program, most tasks have been organized into functions and subroutines for clarity and programmation efficiency. Let it be said now that most of the code outputs are generated by the main program unit. Because a special effort has been made to systematically comment the Fortran source files, there is no point in repeating the code listing in this manual, and the user is strongly encouraged to read the adjoining report and have a closer look at the **hptam.for** hard copy to gain a clear understanding of the organization of the program and of the numerical algorithm used. To improve the readability of the source, variable names have been chosen so as to be (we hope) meaningful to the reader, and have been kept unchanged in the various program units that use them, whenever possible. A detailed description of the input and output variables is given at the beginning of most of the functions and subroutines of the program. There are only two exceptions, the main program unit and the subroutine **inputfile**() in the source file **subnew.for**. However, a description of the variables used in these units can be found in other subroutines, by performing a simple editor search command.

The source file **subnew.for** contains most of the subroutines necessary to calculate the coefficients of the enthalpy and momentum discretized conservation equations. Also, it contains the subroutine **inputfile**() which reads the input file **hp.inp**, echoes it in the output file **hp.vrf** for verification, generates the mesh of the numerical domain and checks if the dimensions of the arrays are suitable for the mesh. The next subroutine, **geometric**(), calculates all the other geometrical parameters necessary for the computation, such as volumes and cross-sectional areas of the numerical cells. Finally, the values of all the physical quantities (such as pressures, temperatures, flow field, etc...) must be known at a given time before a transient calculation can be started or continued, and this initialization is performed by the subroutine **initialize**().

All the physical properties of the working fluids and structural materials of the heat pipe are in the form of functions or subroutines, and are collected in the source file **subold.for**. The code includes the properties of lithium, sodium, potassium and water for the working fluid of heat pipes. The properties of the solid and liquid phases of gallium and tin are also incorporated to perform freeze-and-thaw calculations in an open cavity or porous medium, in the presence of natural convection.

The source file **subold.for** also contains the direct Gauss-elimination Solver for the solution of the pressure elliptic linear system, **solve**(), the subroutine **iternorm**() which calculates the minima and maxima of the physical quantities (such as pressure, temperature, mass fluxes and void fraction) throughout the numerical domain and displays the results in the output file **hp.out**, and the subroutine

graph() which draws the numerical domain and plots the flow fields and frozen fractions in a format readable by TELLAGRAPH on VMS machines (files **hpg.dot** and **hpu.dot**). Because the diameter of the heat pipe is much smaller than its length, the subroutine **graph**() scales the radial direction by the factor **SCALEL** specified in the input file **hp.inp**.

The last source file of the code, **poisson.for**, contains only one program unit, the subroutine **spoisson**(). This routine, the heart of the heat pipe code, forms the pressure corrector step of the solution procedure, and was isolated to simplify and maximize the efficiency of the debugging process. In the pressure corrector step, a combination of the mass balance equations and of an approximation of the momentum conservation equations is used to formulate the Poisson equation, which is solved for the pressure corrections in the vapor and liquid phases. The pressure gradient in the conservative forms of the momentum equations is discretized implicitly. The off-diagonal velocity corrections appearing in the diffusion/convection terms are equated to the diagonal velocity correction, as it is done in the SIMPLE-Consistent algorithm of van Doormaal and Raithby. The evaporation, sublimation, condensation and resolidification mass rates are implicitly written in terms of pressures using the kinetic theory of gases. By eliminating the advanced-time mass fluxes appearing in the mass balance equation and considering the density variation with pressure only, the continuity equation is reduced to a Poisson equation, which is then solved for the pressure corrections. During this corrector step, the volumes of the wick and vapor interfacial cells are treated as functions of the radii of the liquid-vapor (or liquid-solid) interfaces and of the vapor void fractions in the wick pores. These void fractions are functions of the cosines of the contact angles of the liquid meniscus. The latter are implicitly related to the liquid and vapor pressures at the liquid-vapor interface through the radial momentum jump condition (the capillary relationship of Pascal). In the subroutine **spoisson**(), internal iterations are performed to insure convergence of the radii of the liquid-vapor (or liquid-solid) interfaces and of liquid pooling mass rates in the vapor core.

This ends the description of the Fortran source files of the code. It is recommended that the user read the adjoining report and spend some time browsing through the listing of the four files **hptam.for**, **subnew.for**, **subold.for** and **poisson.for** to gain a clear understanding of the organization of the program and of the numerical algorithm used. In the following section, the parameters in the input file **hp.inp** will be described in detail.

3. THE INPUT FILE, "hp.inp"

Every time the code HPTAM is executed, whether starting a new calculation or continuing a transient calculation, the program calls the subroutine **inputfile**(). This subroutine reads the input file **hp.inp**, echoes it in the output file **hp.vrf** for verification, generates the mesh of the numerical domain and checks if the dimensions of the arrays are suitable for the mesh. A sample of a typical input file **hp.inp** (for the startup from a frozen state of a water heat pipe) is included in this package (see also APPENDIX A). The parameters of the input file are described in this section. Note that all physical values in the code are expressed in SI (System International) units. As a convention, every numerical slot of the input file must contain a value, whether it is relevant to the current calculation or not. For example, a value must be given for the convective heat transfer coefficient in the condenser section, even if the heat pipe is radiatively cooled. Therefore, the user can easily create a customized input file from the sample input file given, by simply replacing the existing numerical values by his/her own where relevant. Do not write over the star "*" in column 11, and always insure that the numerical values are preceded and followed by at least one blank space. Finally, when specifying the thermal boundary conditions, a value must be given for every boundary cell of a given axial section. For example, when specifying the heat flux along the evaporator wall, the user must specify 15 values if there are 15 cells in the evaporator region. This feature allows the user to specify a non-uniform flux distribution, for example. After the user has created a new input file, it is possible that the first run of the code will abort, due to a formatting error. In such case, the user can edit the file **hp.vrf**, and find out which one of the above rules was broken.

The second line of the input file is a comment line that can be used to describe the type of calculation performed. This line (except for the first star "*") will be used as a title for all the output files and plots generated by the code HPTAM.

3.1. Initialization and Execution Parameters

The parameter **IFILE** must be set to **zero** when starting a new calculation. In such case, the velocities are set to zero, the initial vapor, liquid (or solid) and wall temperatures are uniform and equal, the vapor pressure is equal to the saturation pressure of the working fluid calculated at the heat pipe temperature, and the liquid pressure (when the working fluid is in the liquid state) is set according to the capillary relationship of Pascal. If no axial nor centripetal acceleration exists, the transient calculation is started right away. After the specified number of time steps have been performed, the main program unit (see **hptam.for**) writes the time of the transient and the values of all the physical quantities (such as pressures, densities, temperatures, mass fluxes, fluid frozen fractions, etc...) at this particular time into the file **hp.sto**. However, if a gravity or acceleration field is present, the code performs internal iterations to calculate the associated non-uniform pressure field and vapor pore void fractions. During this operation, the temperatures (uniform) and velocities (nul) are kept unchanged. After convergence of the pressure field, the time of the transient is set to zero and the values of all the physical quantities are written in the output file **hp.sto**.

It is then possible to continue the transient calculation. This can be done by renaming the file **hp.sto** into **hp.ini**, and setting the parameter **IFILE** to **1** in the input file **hp.inp**. In this case, the time of the transient and the values of all the physical quantities at that particular time are read from the input file **hp.ini**, and the code continues the transient calculation. After the specified number of time steps have been performed, the new time of the transient and the values of all the physical quantities at this time are written into the output file **hp.sto**. This feature of the code allows the user to generate a series of chained runs, and gives him/her enough flexibility to store the values of the physical quantities at the different times of interest during the transient.

The next two parameters control the output of the code. When **IFLAG06**=1, the output files ***.dat** and ***.fld** are generated by the code, as well as the files **hp.out** and **residu.out**. A detailed description of these files is given in the following section of the manual. It is recommended that this option be chosen, as it allows a quick check of the numerical calculation, with a small impact on the CPU time when the number of time steps is greater than 10. The flag **IFLAG13** is normally set to 0. It is set to 1 only during the debugging process of the code. In that case, the code generates additional ***.out** output files (**liqvap.out**, **prop.out**, and **hjump.out**), allowing the user to check the values of chosen variables during the calculation. These output files can be easily modified by the user as the need arises.

The next parameters are the numerical time step **TAU** (in seconds) and the total number of time steps calculated by the code, **ITERMAX**. HPTAM utilizes an iterative fully-implicit numerical discretization scheme to resolve the non-linear couplings of the governing equations. As the time step is increased, more internal iterations must be performed in order to achieve convergence. Therefore, there is an optimal time step which minimizes the amount of CPU time to model a given time interval of the transient. The optimum time step is limited by the vapor flow and normally ranges between 5 and 100 ms for a water heat pipe, and between 25 and 500 ms for a liquid-metal heat pipe. Much larger time steps (on the order of several seconds) can be used when HPTAM models liquid-solid phase-change of fluids in rectangular or cylindrical cavities. The information given in the output files **hp.out** and **residu.out** can be used to evaluate the suitability of the time step for a given problem and particular period of the transient (see the following section). Because the inputs and outputs of HPTAM are organized to allow the user to generate a series of chained runs, there is no advantage in performing a very large number of time step iterations per run. Indeed, a reasonable value of **ITERMAX** is recommended (between 20 and 100) which gives a CPU time per run ranging between 4 and 20 mins. This would allow the user to check the progression of the numerical calculation live on multi-user machines (UNIX and VMS operating systems), without interrupting it. Also, it would not be necessary to rerun the complete calculation again in case a crash of the machine occurred. Instead, only the last 4 to 20 mins of computing would be lost, and the calculation could be continued from the last **hp.sto** output file generated, after the machine were rebooted.

The parameter **EPSNORM** is used to check if steady-state operation has been achieved. The subroutine **iternorm**() calculates the maximum relative changes of the pressures, temperatures, mass fluxes and void fractions in the numerical domain between two time steps, and adds these changes in the variable **SUMNORM**. The main program unit (**hptam.for**) compares this variable with **EPSNORM**. Usually, this parameter is set to a very small value, as the user can use the various plots generated by the code to decide for himself/herself if steady-state operation is achieved or not.

The next parameters are the initial temperature of the heat pipe (**Tinit**), initial interfacial vapor pore void fraction in the wick (**VOIDinit**) and initial vapor-solid or vapor-liquid interfacial radius (**RINTinit**), and are used only when **IFILE=0**. When **Tinit** is below the fusion temperature of the working fluid, **Tmelt**, the fluid is frozen in the wick and **VOIDinit** must be set to 0. When **Tinit>Tmelt**, **VOIDinit>0** and **RINTinit** must be greater than or equal to the radius of the wick surface (or vapor core), **R(Nv)**.

3.2. Definition of Fluid, Materials and Geometry

The types of working fluid (**iwf**), wall material (**iWALL**) and wick material (**iWICK**) are specified next. The code includes the properties of lithium, sodium, potassium and water for the working fluid of heat pipes. The properties of the solid and liquid phases of gallium and Tin are also incorporated to perform freeze-and-thaw calculations in an open cavity or porous medium, in the presence of natural convection; these fluids, however, cannot be used in a heat pipe as properties of their vapor phases are not included in the code. If the user wishes to use a different working fluid, he/her can substitute the properties of the fourth fluid (**iwf=4**), the one referred to as "mercury", in the source file **subold.for**.

In such case, the molecular weight, molecular diameter, fusion temperature and heat of fusion of the working fluid must be specified in the DATA statements **MOLWEIGHT**(4), **MOLDIAM**(4), **Tm**(4), **Hfus**(4) of the subroutine **inputfile**(), in the source file **subnew.for**. Of course, it is always possible to add new working fluids (**iwf**=8, **iwf**=9,...), if necessary.

The wetting angle (**WETTING**) between the liquid and the wick material is specified in degrees. The code insures that the value is greater than 0.5 degree, to avoid the asymptotic behavior of the cosine of contact angle of the liquid meniscus as a function of the interfacial vapor pore void fraction, as the void fraction goes to 1.

The parameter **iGEO** specifies the type of geometry of the physical domain. A cylinder is modeled when **iGEO**=1 (r=0 is the pipe centerline), a symmetric slab when **iGEO**=2 (r=0 is the axis of symmetry) and a non-symmetric slab when **iGEO**=3 (in that case, thermal boundary conditions are applied along the side r=0).

The next parameter, **GRAVITY** (positive), is the magnitude of the gravity field or acceleration. The heat pipe inclination (**tetaGR**, in degrees) is the angle between the horizontal and the heat pipe axis. The angle is positive (0<**tetaGR**<90 °) when the condenser is above the evaporator, negative otherwise. These two parameters are used to calculate the axial component of the acceleration along the heat pipe. The radial component is set to zero for symmetric geometries (**iGEO**=1 or 2) but is non-zero in the case of a tilted slab (**iGEO**=3). In the code, the internal parameter **iGRflag** is set to 1 when **GRAVITY** is non-zero and **IFILE=0**, to bypass enthalpy and momentum calculations and calculate the non-uniform pressure field associated with the gravity field.

3.3. Discretization of Physical Domain

The discretization of the physical domain is defined by the next two sets of parameters. The physical domain is divided into a two-dimensional grid (r,z) of (**Nr**+1) by **Nz** cells. In the axial direction, the evaporator section extends from (j=1) to (j=**Nevap**), the adiabatic section from (j=**Nevap**+1) to (j=**Nadia**), and the condenser section from (j=**Nadia**+1) to (j=**Nz**). The only distinction between these three axial sections arises from the type of thermal boundary condition applied at the outer wall. The mesh generator assumes a uniform mesh size along the adiabatic section. The last evaporator cell and first condenser cell have same axial size as the adiabatic section cells, and geometric progressions are used along the evaporator and condenser to fit the specified number of cells along these regions. In case there is no adiabatic section (the user sets the length and the number of cells in this section to zero), the mesh generator assumes a uniform mesh size along the evaporator, and the first condenser cell has same size as the evaporator cells. For symmetric geometries (**iGEO**=1 or 2), only half of the vapor core is modeled with cells (i=1) through (i=**Nv**). The wick region is represented by cells (i=**iL**=**Nv**+1) through (i=**NL**), and the wall is modeled with cells (i=**NL**+1) through (i=**Nr**). If the condenser section of the heat pipe is convectively cooled, the coolant flow in the (water) cooling jacket is represented by a column of cells (i=**Nr**+1) along the condenser wall. If the number of cells in the vapor region and the thickness of that region are set to 0, the code assumes that the wick region extends from (i=1) to (i=**NL**), and the problem reduces to that of phase-change of a working fluid in a cavity (porous medium or open cavity) in the presence of natural convection. In case of excessive distortion of the numerical mesh, the subroutine **inputfile**() will display suitable messages and the execution of the code will be stopped. Note that **Nu** must be set to **one** in order to activate the Free-Molecule and transition Flow regimes for liquid-metal heats pipes. This is because the non-continuum Flow regimes are modeled using a one-dimensional vapor model (based on the Dusty Gas Model).

3.4. Characteristics of Porous Wick of Heat Pipe

The characteristics of the porous wick are given in the next block of parameters. The wick can be a wire-screened mesh (**iWICKtype**=1), an isotropic porous medium such as a powder or a bed of spheres (**iWICKtype**=2), or an open annulus (**iWICKtype**=3) separated from the vapor core by a thin sheet (with small holes to provide capillary forces). The user must specify the volume porosity of the wick (**Ewick**) and the surface porosity at the vapor-wick interface (**EsWICK**). The later is set equal to **Ewick** for an isotropic porous medium, and is calculated by the code for a wire-mesh. The properties of the wire-mesh depend on the mesh number (**MESHn**, per inch) and on the screen wire diameter (**DiamW**). If the wire diameter is not known, the user can specify a negative value; in that case, the code uses a correlation developed in this work, which gives the wire diameter as a function of the mesh number, based on values commonly used by screen wick manufacturers. The effective pore radius at the vapor-wick surface (**Rpore**) must be specified by the user for an isotropic porous medium or an open annulus. Also, the permeability is specified for an isotropic porous medium; however, it is calculated by the code for the other wick geometries (it is infinite for an open annulus). Finally, the evaporation accommodation coefficient (**ACCOMevap**) which appears in the kinetic theory relationship ranges between 0 and 1. For a properly cleaned and evacuated heat pipe, a value of 1 is achievable and suitable.

3.5. Specification of Thermal Boundary Conditions

The following sections of the input file **hp.inp** deal with the thermal conditions at the outer boundaries of the domain. Let it be recalled that both ends of the heat pipe are assumed thermally insulated. Thermal boundary conditions must be specified along the vertical boundaries of the domain, r=**R**(**Nr**) at the outer pipe wall, and r=0 when dealing with a non-symmetric slab geometry (**iGEO**=3). There are four independent sets of parameters organized under the same model, one for each of the four boundary regions: vertical boundary r=0, evaporator and condenser outer walls, and outer wall of adiabatic section. In each region, isoflux, isothermal, radiative or convective thermal boundary conditions can be specified, independently. One value of the heat flux, or the wall temperature, or the product of the wall emissivity and view factor, or the fluid bulk temperature in the jacket must be given for each numerical cell along the boundary of interest. This feature allows the user to specify non-uniform thermal boundary conditions along the wall of the heat pipe. Note that it is perfectly possible to specify a convective cooling boundary condition along the section referred to as "evaporator" by the code, which extends from (j=1) to (j=**Nevap**). Even if your geometry is symmetric (**iGEO**=1 or 2), you must specify **Nz** values along the vertical boundary r=0. The code will automatically generate symmetric boundary conditions along this boundary, but the subroutine **inputfile**() will always attempt to read **Nz** values into the array **BCWAL0**(). In the input file provided in this package, **Nz**=50, and 50 zero flux values are given along the centerline of the heat pipe (r=0).

When the heat flux is specified along the evaporator outer wall, the code assumes that the heat flux varies from 0 (at time **TIME0**) to the given maximum value $Q_r''(\infty)$, with an exponential period **TIEVAP**:

$$Q_r''(t) = Q_r''(\infty)\left[1 - e^{-\frac{t-\text{TIME0}}{\text{TIEVAP}}}\right] . \qquad (1)$$

The intermediate section of the heat pipe is labeled "adiabatic" as it is in most cases thermally insulated. If there is no adiabatic section (when the user sets the length and the number of cells in this section to zero), one data value is read by the subroutine **inputfile**(), and at least one value (on one line) must be present in the input file. When the heat flux is specified along the condenser outer wall, the code assumes that the heat flux varies from 0 (at time **TIME0**) to the given maximum value $Q_r''(\infty)$, with an exponential period **TICOND**:

$$Q_r''(t) = Q_r''(\infty)\left[1 - e^{-\frac{t-TIME0}{TICOND}}\right] . \qquad (2)$$

If the condenser section of the heat pipe is convectively cooled, the treatment is slightly different. The user must supply the inlet temperature and the mass flow rate of the cooling fluid (water) in the jacket. The coolant flow in the (water) cooling jacket is represented by a column of cells (i=**Nr**+1) along the condenser wall, and the code solves the energy conservation equation for the bulk temperature of the cooling fluid in this region, coupled to the wall of the heat pipe through a convective heat transfer relation.

The environment temperature **Tspace** (necessary for the radiative boundary condition) and the convective heat transfer coefficients for every boundary region are specified next.

3.6. Physical Model and Pressure Sink Parameters

The parameters of the next block control the physical model used in HPTAM. When **iSATURv**=1, the vapor temperature is evaluated along the saturation line as a function of vapor pressure. This option is suitable for low-temperature working fluids (such as water), for the continuum vapor flow regime of high-temperature working fluids, and to some extent to the free-molecule and transition flow regimes of high-temperature working fluids. When **iSATURv**=0, the vapor temperature is obtained from the solution of the enthalpy conservation equation. The divergence of the velocity field and the viscous dissipation in the vapor and liquid phases can be forced to zero by the user. Finally, the flag **iFREEmol** controls the activation of the original calculation model for free-molecule and transition flow regimes. Note that options **iSATURv**=0 and **iFREEmol**=1 have not been tested in this version of the code.

The density of the liquid phase in HPTAM is function of both temperature and pressure, to account for thermal expansion and compressibility of this phase, and model buoyancy-induced convection. Since the physical domain modeled in HPTAM is confined (i.e. of constant volume), changes in temperature will induce changes in densities, accompanied with related pressure changes. This will occur for the case of natural convection of liquid or liquid-solid phase-change in a closed cavity (when the number of cells in the vapor region and the thickness of that region are set to zero) or for the case of a partially frozen heat pipe, when the liquid fluid does not have access to the vapor core (when the wick interface is still frozen). As a fully-thawed heat pipe is slowly and uniformly cooled below the fusion temperature of the working fluid, the fluid would start freezing at the wall first, and would eventually freeze uniformly in the wick, without voids in its bulk. When heat is applied to the evaporator of a frozen water heat pipe, for example, the liquid formed at the wall will be under tremendous tension, since the density of water increases at constant pressure upon melting. Eventually, a void will form when the tension is exceeded in the liquid. In the case of a liquid-metal heat pipe, the density of working fluid decreases upon melting, and the expanding liquid would apply tremendous pressures upon the wick, eventually causing cracks to release its excess volume into the vapor core. To avoid the complications of void formation or cracking in the wick, the model assumes that the numerical cell (**iSINK**, **jSINK**) of the wick is connected to a constant-pressure tank of liquid working fluid (referred to as the "pressure sink"). The tank pressure (**P0SINK**) and the hydrodynamic characteristics of the connecting tube are specified by the user. To simulate a flow or phase-change experiment in a cavity open to air, **P0SINK** should be set to atmospheric pressure. The location of the pressure sink should be the hottest spot of the domain, such as the wick cell adjacent to the heat pipe wall at the evaporator end during the startup from a frozen state. The model automatically disconnects the pressure tank as the melting font reaches the vapor-wick interface in the heat pipe evaporator. The amount of fluid taken from the tank (water heat pipes) or transferred to the tank (liquid-metal heat pipes) during the startup from a frozen state is only a small fraction of the amount of working fluid in the heat pipe, but is accounted for nevertheless in the fluid inventory, since the model conserves the mass. When HPTAM

is used to calculate the transient operation of an initially fully-thawed heat pipe, no pressure sink is necessary and the flag **iFLAGsink** should be set to zero by the user. In that case, the volume of excess liquid is accumulated in the liquid pool at the end of the condenser during heatup, or the level of liquid recedes in the wick during the cooldown of the heat pipe.

3.7. Numerical Solution Parameters

The parameters in the next block of the input file control the efficiency of the numerical solution in terms of computation time. Care must be taken in changing them as they have been optimized based on previous experience and test runs. In this new version of the code, the banded version of the Gauss-elimination solver is only used to solve the elliptic Poisson equation, and includes partial pivoting and row-normalization options. A banded linear equation solver is organized around a data structure that takes advantage of the many zeros in the pentadiagonal matrix of the linear system. The bandwidth of the matrix is preserved as long as no pivoting is performed. Such permutation of line was never necessary in solving the present heat pipe problem, so that the pivoting flag **IPIVOp** is set to 0. The SIMPLEC approximation of the momentum conservation equations to formulate the Poisson equation gives a much faster convergence rate than the SIMPLE approximation, and the former is recommended in HPTAM (**iSIMPLE**=1). Also, implicit discretization of the kinetic theory relationship is essential for the convergence of the algorithm. The evaporation and condensation rates (or sublimation and resolidification rates, as relevant) appearing in the interfacial energy jump condition are linearized in terms of the temperatures during the discretization of the enthalpy conservation equations (**iopENTH**=1); also, these rates are linearized in terms of vapor pressure during the discretization of the mass balance in the vapor region and formation of the elliptic Poisson equation (**iopPOISS**=1).

The iterative Strongly-Implicit Solver **SIS**() is used to solve the 5-point momentum and enthalpy linear systems of equations. The solver, which uses the LU decomposition of Lee combined with the iterative method of solution devised by Stone, has the advantage of not requiring a partial cancellation parameter, as most of the implicit iterative solvers do. Also a SOR (strongly Over-Relaxation) factor of unity is found to be the best choice for the convergence of the iterations. The iterative algorithm of Stone has the disadvantage of using more computational time than Lee's, since it requires calculation of the residual vector at the end of every iteration. However, knowledge of these residuals permits to closely control the number of internal iterations of the numerical procedure, resulting in an overall saving in CPU time. The SIS solver is so efficient for the solution of the enthalpy and momentum linear systems that every internal iteration reduces the maximum residual by at least one order of magnitude, that is, 4 to 6 solver internal iterations (**ITERSMAX**) are sufficient to reduce the residuals by 6 orders of magnitude (**RESEPS**=10^{-6})! The parameters **ITERSMAX** and **RESEPS** are set directly in the main program unit (file **hptam.for**) before the **CALL SIS**() instructions.

To resolve the couplings and non-linearities of the governing equations, HPTAM uses an iterative solution procedure consisting of several sequential steps:

(a) *enthalpy predictor step:* best estimates of pressures and convective fluxes are used explicitly, frozen fractions in the wick are linearized in terms of temperatures, and the enthalpy conservation equations are solved for the temperatures. The frozen fractions in the wick are then updated in terms of temperature.

(b) iterations to (a) are performed until temperatures and frozen fractions converge.

(c) *pressure corrector step:* a simplified form of the momentum conservation equations is used to implicitly relate the mass flow rates and pressure gradients (the SIMPLEC approximation is used when **iSIMPLE**=1). The mass flow rates are eliminated in terms of pressures in the continuity equations. The vapor pore void fraction appearing in the interfacial wick cell volumes is geometrically related to the cosinus of contact angle of the liquid meniscus, and the later is expressed in terms of the liquid and vapor pressures using the capillary relationship (or radial momentum jump

condition). The resulting Poisson equation is solved for the pressure field, and the vapor pore void fractions are updated.

(d) *momentum predictor step:* best estimates of the pressure gradients are calculated, and the momentum conservation equations are solved for the velocity field.

(e) iterations to (c) are performed until velocities and pressures converge.

(f) *properties update:* the thermophysical properties and densities are updated.

(g) iterations to (a) are performed until the mass balance is satisfied (that is, the mass residuals of the Poisson equation have been reduced by at least two orders of magnitude).

In the input file **hp.inp**, the user must specify the maximum number of internal iterations and the convergence criteria for all the loops described above. The internal iterations in step (b) are performed a maximum of **ITERhMAX** times, until the enthalpy residuals are below **CVGenth**. In practice, only a very few iterations (3 to 4) are necessary to reduce these residuals by 6 orders of magnitude. The user must insure that **CVGenth** is smaller than the enthalpy residual at the beginning of a new time step by at least 6 orders of magnitude. These residuals are usually on the order of 1 to 100 and can be read in the output file **residu.out**. Next, a maximum of **INTERMAX** iterations are performed in step (e), until the mass balance residuals are below **CVGSIMPL**. These iterations are critical considering that more than 50% of the CPU time used in the calculation is spent by the direct Gauss-elimination solver for the solution of the elliptic Poisson equation, in step (c). A value of **INTERMAX** between 4 and 6 is optimal. The larger the time step, the larger **INTERMAX** must be to insure convergence of the continuity equations, and there is an optimal discretization time step which minimizes the amount of CPU time to model a given time interval of the transient. Finally, a maximum of **IOKMAX** iterations are performed in step (g), until both enthalpy and mass balance residuals have been reduced below their limits, **CVGenth** and **CVGSIMPL**, respectively.

Next, the user must specify the parameter **DDTmelt**, the half-width of the mushy region, which is used by the liquid-solid phase-change model. It is very important that this parameter remains unchanged during the full transient calculation of a given problem. This is because the frozen volume fractions are related to the temperatures by this parameter. To model the phase-change of pure fluids with a single melting point (by contrast with multi-component fluids undergoing phase-change over a finite temperature range), a very small value of the parameter **DDTmelt** must be used, limited only by machine accuracy since temperature differences in the **DDTmelt** range must be read from the input file **hp.ini** and stored into the file **hp.sto**. When double precision accuracy is used, a value as small as 10^{-8} K can be used for **DDTmelt**.

The next block of lines, referred to as "Elementary Check-Up of Heat Pipe Geometry", must be left untouched by the user. These lines are read by the subroutine **inputfile**() and echoed back to the output file **hp.vrf** with information from the mesh generator, such as total number of numerical cells and locations of cell interfaces in both the radial and axial directions. This feature allows the user to verify that the mesh generated by the code conforms with his/her intentions.

3.8. Parameters for Graphic Outputs

The parameters of the last block of the input file **hp.inp** control the graphic outputs of the code. HPTAM generates four different types of plots: (a) plots ***.fld** showing axial (and radial when relevant) distribution of physical quantities in the heat pipe (such as vapor pressure field, for example) at the last time step calculated by the code; (b) plots ***.dat** showing the axial distribution of a given quantity (such as the vaporization/condensation rate, for example) at different times during the transient calculation; (c) plots ***.time** showing the time history of a given quantity at a given location in the heat pipe (such as the temperature in a specified numerical cell); and (d) plots of the discretized domain showing the velocity fields and/or the fluid frozen fractions in the wick (files **hpg.dot** and **hpu.dot**).

Data points are printed in the time history files ***.time** every **iPRINT** time steps. For example, for a time step **TAU**=0.1 s, the parameter **iPRINT** must be set to 10 to print a data point every second of the transient. When a series of chained runs is performed to cover an extended transient period, the files ***.time** generated by every HPTAM run are appended at the end of the files ***.tim0**, which grow as the transient proceeds (this feature is described in more detail in the following section of the manual). The parameter **iPRINT** must be selected based on the characteristic transient period of the heat pipe analyzed, and on the extent of the transient investigated. For example, when a 20-mn real-time transient is modeled and data points are collected every 0.1 s in the output files ***.time**, the files ***.tim0** will contain a total of 10x20x60=12,000 data values each, which is obviously excessive.

Data points are printed in the files ***.dat** every **iPRDAT** time steps. For example, for a time step **TAU**=0.1 s, the parameter **iPRDAT** must be set to 10 to show the axial distribution of given quantities every second of the transient. Usually, the parameter **iPRDAT** is set to a fifth (or a tenth) of **ITERMAX** to display a total of 5 (or 10) curves in the ***.dat** plots. By contrast with the history files ***.time**, the plots ***.dat** are not appended as a series of chained runs is performed, and they stand by themselves.

Next the user must specify the location in the numerical domain of the liquid and vapor pressures probes (history files **pressl.time** and **pressv.time**), the temperature probe (file **temp.time**), the radial mass flux probe (file **radial.time**) and the axial velocity probe (file **axial.time**). In the input file provided in this package, the liquid pressure and temperature probes are located in the wick cell adjacent to the heat pipe wall, in the middle of the evaporator, the vapor pressure and axial velocity probes are located along the centerline of the vapor core, at the evaporator end and the exit of the adiabatic section, respectively, and the radial mass flux probe is located at the wick-vapor interface, at the evaporator end, so that the plot **radial.tim0** will show the transient variation of the vaporization rate at this location.

The parameters **SCALEL** and **ARROWL** are used by the subroutine **graph**(), which draws the numerical domain and plots the flow fields and frozen fractions in a format readable by TELLAGRAPH on VMS machines (files **hpg.dot** and **hpu.dot**). Because the diameter of the heat pipe is much smaller than its length, the subroutine **graph**() scales the radial direction by the factor **SCALEL**. The parameter **ARROWL** controls the size of the velocity arrows, with respect to the smallest cell dimension in the numerical domain. A value of 3 usually generates clear and elegant arrow flow fields.

Finally, the last parameter, **IPLOTPC**, specifies the operating system available to the user for plotting the results. The output files ***.fld**, ***.dat** and ***.tim0** can be generated in a form compatible with the TELLAGRAF or EASY–PLOT software packages, whether a VMS operating system machine or an IBM-compatible Personal Computer (running MS-DOS) is available to the user. In case the code is running on a UNIX machine, it is necessary that a VMS machine or a MS-DOS machine be used to visualize the plots generated by the code. For example, the File Transfer Program (FTP) can be used to transfer the output files generated on the UNIX machine to the MS-DOS machine via the high-speed Ethernet network, in order to visualize these plots with the EASY–PLOT software. Note that the plots **hpg.dot** and **hpu.dot** can only be visualized by the TELLAGRAF software, whereas the output files **twall.tim0** and **void.tim0** are compatible with the EASY–PLOT software, only. Of course, it is always possible for a user familiar with a different plotting package to rewrite the outputs of the code in the new format. All the output files are generated by the main program unit (file **hptam.for**), except the plots **hpg.dot** and **hpu.dot** which are created by the subroutine **graph**() in the source file **subold.for**.

This ends this section on the description of the parameters of the input file **hp.inp**. In the next section, the input and output files of the code HPTAM are reviewed and described.

4. INPUT AND OUTPUT FILES FOR HPTAM

During the execution of the code HPTAM, various output files are generated. The file **hp.vrf** can be visualized to verify that the code has read the input file **hp.inp** correctly. After calculation of the required number of time steps, HPTAM writes the time of the transient and the values of all the physical quantities at this time in the file **hp.sto**. This file can be renamed as **hp.ini** and used by HPTAM as input to continue the transient calculation. The code also generates the output files **hp.out** and **residu.out** which contain information pertaining to the convergence of the numerical procedure. Finally, HPTAM generates various output files for plotting the results. All these output files are described in details in the following subsections.

4.1. Input And Output Files For Running The Code

Every time the code HPTAM is executed, whether starting a new calculation (**IFILE=0**) or continuing a transient calculation (**IFILE=1**), the subroutine **inputfile**() reads the input file **hp.inp** and generates the mesh of the numerical domain. A sample of a typical input file **hp.inp** (for the startup from a frozen state of a water heat pipe) is included in this package (see also APPENDIX A). The parameters of the input file are described in the previous section of this manual.

The subroutine **inputfile**() also generates the verification file **hp.vrf**, a duplicate of the input file **hp.inp**, which contains values of the input parameters as read and understood by HPTAM (the hydrodynamic properties of the porous wick displayed in **hp.vrf** are that calculated by the model). Finally, the subroutine **inputfile**() provides additional information into the output file **hp.vrf**, such as total number of numerical cells and locations of cell interfaces in both the radial and axial directions. This feature allows the user to verify that the mesh generated by the code conforms with his/her intentions. After the user has created a new input file **hp.inp**, it is possible that the first run of the code will abort, due to a formatting error. In such case, the user can edit the verification file **hp.vrf**, and find out which one of the formatting rules was broken (these rules are described in the previous section of the manual).

The parameter **IFILE** must be set to **zero** when starting a new calculation. If no external acceleration exists, the transient calculation is started right away. After the specified number of time steps has been performed, the main program unit (see **hptam.for**) writes the time of the transient and the values of all the physical quantities at this particular time (such as pressures, densities, temperatures, mass fluxes, fluid frozen fractions, etc...) into the file **hp.sto**. However, if a gravity or acceleration field is present, the code performs internal iterations to calculate the associated non-uniform pressure field and vapor pore void fractions. During this operation, the temperatures (uniform) and velocities (nul) are kept unchanged. After convergence of the pressure field, the time of the transient is set to zero and the values of all the physical quantities are written in the output file **hp.sto**.

It is then possible to continue the transient calculation. This can be done by renaming the file **hp.sto** into **hp.ini**, and setting the parameter **IFILE** to **1** in the input file **hp.inp**. In this case, the time of the transient and the values of all the physical quantities at that particular time are read from the input file **hp.ini**, and the code continues the transient calculation. After the specified number of time steps has been performed, the new time of the transient and the values of all the physical quantities at this time are written into the output file **hp.sto**. This feature of the code allows the user to generate a series of chained runs, and gives him/her enough flexibility to safeguard the values of the physical quantities at the different times of interest during the transient. In addition, the user has the capability of checking the progression of the numerical calculation live on multi-user machines (UNIX and VMS operating systems), without interrupting it. Also, it would not be necessary to rerun the complete calculation again in case a crash of the machine occurred. Instead, only the last code run results would be lost, and the calculation could be continued from the last **hp.sto** output file generated, after the machine were rebooted.

4.2. Output Files "*.out" For Checking Convergence and Debugging

When **IFLAG06**=1, the output files **hp.out** and **residu.out** are generated by the code. The information given in these files can be used to check the convergence of the numerical procedure and evaluate the suitability of the time step for a given problem and particular period of the transient. For every time step, the output file **hp.out** lists the steps of the numerical procedure performed by the code (these steps are described in the previous section of the manual). The numerical solution of heat transfer and fluid flow problems can be advanced in time by using explicit or implicit discretization schemes. Explicit schemes are easily programmed but have a severe stability restriction on the time step ($\Delta t < \Delta t^*$), which may compromise their efficiency. The explicit limit Δt^* is defined as:

$$\Delta t^* = \mathrm{MIN}\left\{\Delta t^*_{i,j}\right\}, \tag{3}$$

where $\Delta t^*_{i,j}$ is the local explicit time step limit, which depends on the numerical mesh size (van Doormaal and Raithby 1984). In HPTAM, the factor E is defined by:

$$E_{i,j} = \frac{\Delta t}{\Delta t^*_{i,j}}\ , \tag{4}$$

and the range of this parameter (as well as the location in the numerical mesh of its maximum value) is given in the file **hp.out** for the enthalpy and momentum linear systems of equations. Of particular interest are the maximum value E_{max}, which has the expression:

$$E_{max} = \mathrm{MAX}\left\{E_{i,j}\right\} = \mathrm{MAX}\left\{\frac{\Delta t}{\Delta t^*_{i,j}}\right\} = \frac{\Delta t}{\Delta t^*}\ , \tag{5}$$

and the dominance of the discretized system of equations, which is defined as $(1+E_{max})^{-1}$. The larger the time step Δt, the smaller the dominance of the system of linear equations. Note that a time step Δt equal to the characteristic time step Δt^* corresponds to E_{max}=1 and is the maximum stability limit for the explicit discretization scheme. The implicit discretization scheme used in HPTAM allows the use of values of E_{max} in excess of 10 for fast transient calculations. In practice, the time step is limited by the discretization of the momentum conservation equations in the vapor (where transport by convection dominates the transport by diffusion) and is optimum when the values of E_{max} for these equations range between 15 and 30 (that is, the implicit discretization scheme used in HPTAM allows the use of time steps as large as 30 times the maximum time step for the explicit discretization scheme!). The value of E_{max} for the discretized enthalpy equations in the liquid and solid phases is never a concern (as transport by diffusion dominates the transport by convection), even when solid-liquid phase-change occurs. It is not unusual to find values of E_{max} as high as 300 for the enthalpy equations.

At the end of every time step, the minimum and maximum values of each physical quantity (temperature, pressure, void fraction, and axial and radial mass fluxes) throughout the numerical domain are displayed in the file **hp.out** (as well as the location of the extrema in the numerical mesh). In addition, the subroutine **iternorm**() calculates the maximum relative changes of the same physical quantities between two time steps, and adds these changes in the variable **SUMNORM**. Care must be taken when interpreting the value of **SUMNORM**, because large values of the relative changes in the mass fluxes are characteristics of a negligible local mass flux, and not of a diverging flow field (for example, the radial mass flux at the center of a vortex can be negligible and oscillate between 10^{-5} and 10^{-6} times the maximum mass flux in the domain, resulting in relative changes as large as 1,000 % !). Finally, the input power to the evaporator (**PWREVAP**), and the output powers (**PWRINTER**,

PWRCOND, PWRJACK) at the adiabatic section, condenser section and cooling jacket (when relevant) are printed in the file **hp.out** in watts. Additional information is also given on pressure sink and pooling of liquid in the vapor core when relevant. When the flag **IFLAG13** is set to 1 (during the debugging process of the code), every major subroutine called by the code displays a message in the output file **hp.out**. This feature generates a very detailed listing of the operations performed by HPTAM, and allows the user to check the numerical scheme during the debugging process.

The residual file **residu.out** contains information on the convergence of the internal iterations of the numerical procedure. The various sequential steps of the iterative solution procedure were described in detail in the previous section of this manual. In the input file **hp.inp**, the user must specify the maximum number of internal iterations and the convergence criteria for all the internal loops. A sample of the residuals, as displayed in the output file **residu.out**, is given in Table 1. Note that the maximum residual is the maximum absolute value of the source terms of the discretized equations (when the latter are expressed in terms of corrections) before the system of equations is solved for the correction field (temperature, pressure or velocity correction).

TABLE 1. Sample of residuals as displayed in the output file **residu.out**.

1 *** EPSILON= .689E-12 *** ENTHALPY RESIDU= .974E-01 (15,18)							
2 *** EPSILON= .691E-12 *** ENTHALPY RESIDU= .300E-06 (13,18)							
KINETIC	PASCAL	CONTINUITY	RADIAL M.	AXIAL M.	$\|P'\|_{max}$	$\|\mu_c'\|_{max}$	$\|\dot{m}'\|_{max}$ 13
.120E-12	.165E-03	.132E-07	.908E-09	.285E-08	.798E-01	.884E-05	.359E-06
		(9,10)	(7, 1)	(3,33)	(9, 9)		
.670E-16	.759E-06	.584E-10	.791E-10	.126E-09	.103E-03	.369E-07	.332E-07
		(9,50)	(7,50)	(3,35)	(9,50)		

The first two lines in Table 1 correspond to the enthalpy predictor step (a) and the internal iterations in step (b). The number preceding the stars "***" is the iteration number **ITERh**. EPSILON is the maximum residual of the energy jump condition along the vapor-wick interface, and is normally below 10^{-11}. The maximum source term (or residual) of the enthalpy conservation equations is 0.974×10^{-1} before the solution of the linear system is processed, and occurs in the numerical cell (i,j)=(15,18). After the discretized equations are solved for the temperature corrections using the iterative **SIS**() solver, and after these corrections are applied to the enthalpy conservation equations, the maximum enthalpy residual is reduced to 0.3×10^{-6}. Additional information is displayed by the phase-change model as one or several numerical cells change their types (in such case, these additional lines always start with the word "TRANSIT", for transition). Iterations are performed a maximum of **ITERhMAX** times, until the enthalpy residuals are below **CVGenth**.

Next, the momentum conservation equations are discretized, and the Poisson equation is formed using the SIMPLEC approximation in step (c). The next lines in Table 1 display the residuals of the kinetic theory relationship, of the radial momentum jump condition (or Pascal relationship) and of the mass balance before the Poisson equation is solved for the pressure corrections (the number 13 on the right of the third line of Table 1 is the time step number **ITER**). After the solution of the Poisson equation is obtained using the direct Gauss-elimination solver **SOLVE**(), the maximum pressure correction $|P'|_{max}$, and maximum corrections in the cosinus of the contact angle of the liquid meniscus $|\mu_c'|_{max}$ and in the vaporization/condensation rate at the vapor-wick interface $|\dot{m}'|_{max}$ are computed and displayed in their respective column. Also, the maximum residuals of the radial and axial momentum discretized equations are displayed after the pressure gradients have been updated using the calculated pressure

corrections. Next, the momentum conservation equations are solved for the velocity corrections in step (d), and a maximum of **INTERMAX** iterations to (c) are performed in step (e) until the mass balance residuals are below **CVGSIMPL**. These iterations are critical considering that more than 50% of the CPU time used in the calculation is spent by the direct Gauss-elimination solver for the solution of the elliptic Poisson equation, in step (c). A value of **INTERMAX** between 4 and 6 is optimal. The larger the time step, the larger **INTERMAX** must be to insure convergence of the continuity equations, and there is an optimal discretization time step which minimizes the amount of CPU time to model a given time interval of the transient. Additional information is displayed in the output file **residu.out** by the **spoisson**() subroutine as a wet point appears at the liquid-vapor interface, the contact angle of the liquid meniscus is exceeded at some location, the liquid level recovers in the wick, or a liquid film appears on top of the frozen substrate. In such cases, these additional lines are always preceded with *%%%*, and are easily found by performing a simple editor search command for "%%%".

When the flag **IFLAG13** is set to 1 (during the debugging process of the code), the code generates additional ***.out** output files (**liqvap.out**, **prop.out**, and **hjump.out**), allowing the user to check the values of chosen variables during the calculation. These output files can be easily modified by the user as the need arises.

4.3. Output Files For Graphing Results

HPTAM generates four different types of plots: (a) plots ***.fld** showing axial (and radial when relevant) distribution of physical quantities in the heat pipe (such as vapor pressure field, for example) at the last time step calculated by the code; (b) plots ***.dat** showing the axial distribution of a given quantity (such as the vaporization/condensation rate, for example) at different times during the transient calculation; (c) plots ***.time** showing the time history of a given quantity at a given location in the heat pipe (such as the temperature in a specified numerical cell); and (d) plots of the discretized domain showing the velocity fields and/or the fluid frozen fractions in the wick (files **hpg.dot** and **hpu.dot**).

After the last time step has been calculated, the code generates 14 different output files ***.fld**. A brief description of these files follows next:

grliq.fld	radial mass flux of liquid in the wick;
gzliq.fld	axial mass flux of liquid in the wick;
mach.fld	axial Mach number distribution in the vapor core;
press.fld	total liquid and vapor interfacial pressures and vapor saturation pressure;
pressl.fld	liquid pressure distribution in the wick;
pressv.fld	vapor pressure distribution in the core;
temp.fld	temperature field in the heat pipe (vapor, wick and wall);
templ.fld	temperature distribution in the wick and wall of the heat pipe;
tempv.fld	vapor temperature distribution in the core;
tint.fld	temperatures along the vapor-wick interface;
urvap.fld	radial velocity field in the vapor;
uzvap.fld	axial velocity field in the vapor;
visdl.fld	viscous dissipation in the liquid-wick;
visdv.fld	viscous dissipation in the vapor phase.

The code generates 7 different output files ***.dat**, plotting a curve every **IPRDAT** time steps. A brief description of these files follows next:

evap.dat	evaporation/condensation (or sublimation/resolidification) interfacial mass rates;
flflood.dat	interfacial pooling mass flow rates in the vapor core;
pressl.dat	axial distribution of liquid pressure at the vapor-wick interface;
pressv.dat	axial distribution of vapor pressure along the centerline of the heat pipe;

rint.dat	normalized radius of vapor-fluid interface in the wick, $(R_{wk}\text{-}Rint_j) / \Delta R_{iL}$;
tint.dat	axial distribution of fluid interfacial temperature in the wick;
void.dat	axial distribution of vapor pore void fraction in the wick.

Finally, the code generates 17 different time-history output files ***.time**. A brief description of these files follows next:

axial.time	axial velocity at location (**iUz,jUz**);
flin.time	input power to the wall of the evaporator section;
fljack.time	power carried away by water coolant in condenser cooling jacket;
flout.time	output power to the wall of the condenser section;
llevel.time	normalized radius of liquid level at the evaporator end, $(R_{wk}\text{-}Rint_1) / \Delta R_{iL}$;
mass.time	total mass of working fluid (including liquid pool and pressure sink tank);
mpool.time	mass of working fluid in the liquid pool (at the condenser end);
pressl.time	pressure at location (**iLpress,jLpress**);
pressv.time	pressure at location (**iVpress,jVpress**);
radial.time	radial mass flux at location (**iUr,jUr**);
solmass.time	mass of frozen working fluid in the heat pipe;
tau.time	numerical time step **TAU** (Δt) of the calculation;
temp.time	temperature at location (**iTEMP,jTEMP**);
thick.time	extent (thickness) of the liquid pool (at the condenser end);
tpool.time	temperature of liquid pool;
twall.time	outer wall temperature at several axial locations along the heat pipe;
void.time	vapor pore void fraction at several axial locations along the heat pipe.

When a new calculation is started, the parameter **IFILE** is set to **zero** and the code generates the titles, legends and necessary plot commands in the output files ***.time**. After the first run, a set of commands is executed (described in more details in the following sections of the manual), whose purpose is to prevent filename collisions, rename the file **hp.sto** into **hp.ini**, and rename every ***.time** file into ***.tim0**. It is then possible to continue the transient calculation, after setting the parameter **IFILE** to **1** in the input file **hp.inp**. When the next HPTAM run is executed, the time of the transient and the values of all the physical quantities at that particular time are read from the input file **hp.ini**, and the code continues the transient calculation. After the specified number of time steps has been calculated, HPTAM generates new files ***.time**, which contain only the data points associated with the transient period calculated. Then, a different set of commands is executed, whose purpose is to prevent filename collisions, rename the file **hp.sto** into **hp.ini**, and append every ***.time** file at the end of the associated file ***.tim0**. Therefore, as a series of chained runs is performed to cover a large transient period, the files ***.tim0** grow in size.

By contrast with the history files ***.time**, the plots ***.dat** are not appended as a series of chained runs is performed, and they stand by themselves.

The output files ***.fld**, ***.dat** and ***.tim0** can be generated in a form compatible with the TELLAGRAF (**IPLOTPC**=2) or EASY–PLOT (**IPLOTPC**=1) software packages, whether a VMS operating system machine or a Personal Computer running MS-DOS is available to the user. In case the code is running on a UNIX machine, it is necessary that a VMS machine or a MS-DOS machine be used to visualize the plots generated by the code. For example, the File Transfer Program (FTP) can be used to transfer the output files generated on the UNIX machine to the MS-DOS machine via the high-speed Ethernet network, in order to visualize these plots with the EASY–PLOT software. Because the TELLAGRAF software requires that closing plot commands be present after the data points, the VMS

command file **time.com** must be executed before the ***.tim0** history plots can be visualized on the VMS machine. The **time.com** command simply copies every ***.tim0** file into ***.tim** and appends the file **end.tim** at the end of every ***.tim** file. The file **end.tim** contains the 2 closing TELLAGRAF commands:

```
END OF DATA.
GO.
```

Finally, and to close this section of the manual, the plots **hpg.dot** and **hpu.dot** can only be generated by the TELLAGRAF software, whereas the output files, **twall.tim0** and **void.tim0**, are compatible with the EASY–PLOT software, only.

The next 3 sections of the manual describe how to install and run the code on UNIX, MS-DOS and VMS machines. The following notations will be used throughout: the prompts will be represented by the characters "%", ">" and "$" on UNIX, MS-DOS and VMS machines, respectively. All prompt commands will be typed normally; filenames, variable and subroutine names will be typed in **bold**. Certain filenames and directory names are left to the discretion of the user; in such case, the proposed name in the manual will by typed in ***bold and italic.***

5. RUNNING HPTAM ON UNIX MACHINE

5.1. Installation of HPTAM on UNIX Machine

To begin with, the user is advised to create a new directory (***HPTAM***) on his/her main UNIX account to house the code HPTAM:

% cd (or % cd $HOME)
% mkdir ***HPTAM***
% cd ***HPTAM***

The File Transfer Program (FTP) can be used to transfer the files from the MS-DOS machine to the UNIX directory ***HPTAM*** via the high-speed Ethernet network. After the transfer, the directory includes the four Fortran source files (**hptam.for**, **subnew.for**, **subold.for** and **poisson.for**), a standard input file (**hp.inp**), and several command files to run the code (**gorun**, **showjob**, **clean.com**, **start.com**, **coop.com**, and **job*.com** files). A copy of the command files is provided in APPENDIX B of this manual.

It is recommended to change the prefix (**.for**) of the Fortran source files into (**.f**), as this is the default prefix on the UNIX operating systems:

% mv **hptam.for** **hptam.f**
% mv **subnew.for** **subnew.f**
% mv **subold.for** **subold.f**
% mv **poisson.for** **poisson.f**

Due to the large size of the object files, the executable file of the code and the large number of output files generated by HPTAM, it is recommended to create another directory on temporary disk space (/tmp/***HP_RUN***) to compile and execute the code:

% mkdir /tmp/***HP_RUN***
% cd /tmp/***HP_RUN***
% cp $HOME/***HPTAM***/* .

The user must be aware that any file in temporary space (/tmp) not used or "touched" after a certain period of time (commonly 3 days) will be automatically deleted by the operating system. It is therefore necessary to backup the wanted files onto the main disk space (when memory is available) or on a magnetic tape device, for example. When running a heat pipe calculation for several days, it is possible to reset the clock to zero by "touching" the files:

% touch *

However, this command should not be used to store on temporary space a very large number of files for a long period of time, as such practice is discouraged (sometimes sanctioned) by system operators.

Because the executable mode of command files can be lost when these files are transferred or copied to a different location, it is necessary to reset their mode to executable using the "chmod" (change mode) command:

% cd /tmp/***HP_RUN***
% chmod u+x **gorun**
% chmod u+x **showjob**
% chmod u+x ***.com**

It is now necessary to compile and link the Fortran source files of the code. On an IBM RS-6000 Model 370 machine running IBM AIX Version 3.2 operating system, for example, the "xlf" compiler is used to compile every source file separately:

% xlf –c **hptam.f**
% xlf –c **subnew.f**
% xlf –c **subold.f**
% xlf –c **poisson.f**

This compilation generates the 4 object files **hptam.o**, **subnew.o**, **subold.o** and **poisson.o**. The following command links these object files and creates the executable **hptam**:

% xlf –o **hptam** *.o

Now, the /tmp/***HP_RUN*** directory contains all the files necessary to execute the code.

5.2. Starting a New Calculation on UNIX Machine

Before a new calculation is started, the set of commands **clean.com** must be executed. These commands prevent the possibility of filename collisions by adding a character "1" at the end of the filenames of existing files (the UNIX operating system would not allow the code to create a new file whose name already exists in the directory of interest).

% cd /tmp/***HP_RUN***
% **clean.com**

Next the input file **hp.inp** must be edited. The parameter **IFILE** must be set to **zero** when starting a new calculation. Because the inputs and outputs of HPTAM are organized to allow the user to generate a series of chained runs, there is no advantage in performing a very large number of time step iterations per run. Indeed, a reasonable value of **ITERMAX** is recommended (between 20 and 100) which gives a CPU time per run ranging between 4 and 20 mins. This would allow the user to check the progression of the numerical calculation live on the UNIX multi-user machine, without interrupting it. Also, it would not be necessary to rerun the complete calculation again in case a crash of the machine occurred. Instead, only the last 4 to 20 mins of computing would be lost, and the calculation could be continued from the last **hp.sto** output file generated, after the machine were rebooted.

% vi **hp.inp**

Then, the first run of the code is executed:

% **gorun hptam**

The command **gorun** prevents the possibility of filename collisions by renaming the existing **time.log** and **job.log** files as **time.log1** and **job.log1**, and executes the job **hptam** in the background. The code outputs generated by the WRITE(6,...) Fortran instructions are directed to the output file **job.log**, the /usr/bin/time CPU command is executed, and the commands and CPU time of the job are echoed in the output file **time.log**.

When the parameter **IFILE** is **zero**, the transient calculation is started right away if no external acceleration exists. After the specified number of time steps has been performed, the main program unit writes the time of the transient and the values of all the physical quantities at this particular time into the file **hp.sto**. However, if a gravity or acceleration field is present, the code performs internal iterations to calculate the associated non-uniform pressure field and vapor pore void fractions. After convergence of the pressure field, the time of the transient is set to zero and the values of all the physical quantities are written in the output file **hp.sto**.

The command **showjob** can be used to display the user's jobs running in the background. Note that the name of the user's account (USERID) must be specified in the command file **showjob**.

% **showjob**

The progress of the job can be checked by typing the content of the ***.log** files on the terminal:

% cat ***.log**

After the user has created a new input file **hp.inp**, it is possible that the first run of the code will abort, due to a formatting error. In such case, the user can edit the verification file **hp.vrf** (a duplicate of the input file **hp.inp**, which contains values of the input parameters as read and understood by HPTAM), and find out which one of the formatting rules was broken. Note that additional information is provided into the output file **hp.vrf**, such as total number of numerical cells and locations of cell interfaces in both the radial and axial directions. This feature allows the user to verify that the mesh generated by the code conforms with his/her intentions.

% vi **hp.vrf**

At the end of the first run, the code generates the titles, legends and necessary plot commands in the output history files ***.time**. After the first run, the command **start.com** must be executed, whose purpose is to prevent filename collisions, rename the file **hp.sto** into **hp.ini**, and copy every ***.time** file (actually ***.time1** file at that point) into the time history files ***.tim0**.

% **start.com**

5.3. Continuing a Transient Calculation on UNIX Machine

It is then possible to continue the transient calculation, after setting the parameter **IFILE** to **1** in the input file **hp.inp**.

% vi **hp.inp**

When the next HPTAM run is executed, the time of the transient and the values of all the physical quantities at that particular time are read from the input file **hp.ini**, and the code continues the transient calculation. After the specified number of time steps has been calculated, the new time of the transient and the values of all the physical quantities at this time are written into the output file **hp.sto**, and HPTAM generates new files ***.time**, which contain only the data points associated with the transient period calculated. Then, the command **coop.com** must be executed, whose purpose is to prevent filename collisions, rename the file **hp.sto** into **hp.ini**, and append every ***.time** file at the end of the associated file ***.tim0** (using the UNIX command "cat"). To cover a large transient period, a series of chained runs can be performed by alternating code runs (% **gorun hptam**) with **coop.com** commands (% **coop.com**). The files ***.tim0** grow in size after every HPTAM run.

For convenience, job command files are included in this package (**job0.com**, **job1.com**, **job2.com**, ...), which perform a series of 20 chained runs each. After setting the parameter **IFILE** to **1** in the input file **hp.inp** (the first run to start a new calculation must always be executed by hand with the command % **gorun hptam**), the transient calculation can be continued by executing the first **job0.com** process:

% **gorun job0.com**

This process performs 20 code runs in succession and saves the files **hp.sto** generated by every run as **hp.001**, **hp.002**, **hp.003**, ..., to **hp.020**. It then starts the next process by executing the command % **gorun job1.com**. Seven jobs are chained that way automatically (**job0.com**, **job1.com**, ..., to **job6.com**), for a total of 140 code runs.

During the execution of a large **job*.com** process in the background, the command **showjob** can be used to display the user's jobs and their associated process numbers:

% **showjob**

To interrupt a job running in the background of process number ***process#***, the user can use the "kill" command:

% kill –9 ***process#***

The progress of the job can be checked by typing the content of the ***.log** files on the terminal:

% cat ***.log** |more

or by editing these files:

% vi **job.log**
% vi **time.log**

Note that while the files **job.log** and **time.log** contain information pertaining to the current job being executed in the background (**job3.com** for example), the files **job.log1** and **time.log1** contain information on the previous job (**job2.com**, in that example). These files can be visualized with the commands:

% cat ***.log1** |more
% vi **job.log1**
% vi **time.log1**

When **IFLAG06**=1 in the input file **hp.inp**, the output files **hp.out** and **residu.out** are generated by the code. The information given in these files can be used to check the convergence of the internal iterations of the numerical procedure and evaluate the suitability of the time step for a given problem and particular period of the transient (see Section 4.2 of this manual).

% vi **hp.out**
% vi **residu.out**

Similarly, while the files **hp.out** and **residu.out** contain information pertaining to the current run of the code, the files **hp.out1** and **residu.out1** contain information on the previous HPTAM run. These files can be visualized with the commands:

% vi **hp.out1**
% vi **residu.out1**

5.4. Extracting Data and Plots on UNIX Machine

HPTAM generates four different types of plots (Section 4.3): (a) plots ***.fld** showing axial (and radial when relevant) distribution of physical quantities in the heat pipe at the last time step calculated by the code; (b) plots ***.dat** showing the axial distribution of a given quantity at different times during the

transient calculation; (c) plots ***.time** showing the time history of a given quantity at a given location in the heat pipe; and (d) plots of the discretized domain showing the velocity fields and/or the fluid frozen fractions in the wick (files **hpg.dot** and **hpu.dot**).

The output files ***.fld**, ***.dat** and ***.tim0** can be generated in a form compatible with the TELLAGRAF (**IPLOTPC**=2) or EASY–PLOT (**IPLOTPC**=1) software packages, whether a VMS operating system machine or a Personal Computer running MS-DOS is available to the user. When the code is running on a UNIX machine, it is necessary that a VMS machine or a MS-DOS machine be used to visualize the plots generated by the code. For example, the File Transfer Program (FTP) can be used to transfer the output files generated on the UNIX machine to the MS-DOS machine via the high-speed Ethernet network, in order to visualize these plots with the EASY–PLOT software. Note that the plots **hpg.dot** and **hpu.dot** can only be visualized by the TELLAGRAF software (running on VMS machines), whereas the output files **twall.tim0** and **void.tim0** are compatible with the EASY–PLOT software (running on MS-DOS machines) only. Of course, it is always possible for a user familiar with a different plotting package to rewrite the outputs of the code in the new format. Because the TELLAGRAF software requires that closing plot commands be present after the data points, the VMS command file **time.com** must be executed before the ***.tim0** history plots can be visualized on the VMS machine. The user is advised to read section 7 of this manual for more details on how to plot the code outputs using the TELLAGRAF software package.

As the main heat pipe transient process (**job*.com**) is being executed in the background, the user can begin to extract the data of interest by carrying on simultaneous calculations in a ***DISTILL*** sub-directory. This sub-directory is setup as follows:

```
% cd /tmp/HP_RUN
% mkdir DISTILL
% cp hptam      DISTILL
% cp hp.inp     DISTILL
% cp clean.com  DISTILL
% cp gorun      DISTILL
% cp showjob    DISTILL
% cd DISTILL
% chmod u+x hptam
% chmod u+x clean.com
% chmod u+x gorun
% chmod u+x showjob
```

Let us assume, for example, that the user is interested into the pressure and temperature fields in the heat pipe at time 222 s of the transient. For a numerical time step **TAU**=0.1 s and a total number of time steps **ITERMAX**=50 (these parameters are specified in the input file **hp.inp**), every run of the code progresses the transient calculation by 5 s of real time. In that case, the last run of the process **job1.com** has created the file **hp.040**, which contains the values of all the physical quantities in the heat pipe at time 200 s of the transient. After the process **job2.com** has created the file **hp.044**, which corresponds to the time 220 s, the time of interest (222 s) can be reached by performing 20 additional iterations (with an identical time step of 0.1 s). The file **hp.044** is copied to the sub-directory ***DISTILL***, and the parameter **ITERMAX** is set to 20 in the input file **hp.inp**:

```
% cd /tmp/HP_RUN
% cp hp.044 DISTILL
% cd DISTILL
% vi hp.inp
```

The short job is performed in the sub-directory /tmp/***HP_RUN/DISTILL*** by running the command **clean.com** to prevent possible name collisions, copying the file **hp.044** to **hp.ini**, and running the executable **hptam**:

% **clean.com**
% cp **hp.044 hp.ini**
% **gorun hptam**

The progress of the job can be checked with the commands:

% **showjob**
% cat ***.log**

After the job is executed, the output files **temp*.fld** and **press*.fld** (see Section 4.3) generated by the run contain the temperature and pressure fields in the heat pipe at time 222 s of the transient. To graph these outputs, these files must be transferred to a VMS or MS-DOS machine, as explained before.

6. RUNNING HPTAM ON MS-DOS MACHINE

6.1. Installation of HPTAM on MS-DOS Machine

Included in this package is a 3.5" MS-DOS formatted disk containing the four Fortran source files of the code, along with a standard input file and three sets of command files, one for each operating system (MS-DOS, UNIX and VMS). To begin with, the user is advised to create a new directory (***HPTAM***) on his/her Personal Computer to house the code HPTAM:

> mkdir D:***HPTAM***
> copy b:***.for** D:***HPTAM***
> copy b:**MS_DOS***.* D:***HPTAM***

After the transfer, the directory ***HPTAM*** includes the four Fortran source files (**hptam.for**, **subnew.for**, **subold.for** and **poisson.for**), a standard input file (**hp.inp**), and several command files to run the code (**clean.bat**, **start.bat**, **coop.bat**, and **job*.bat** files). A copy of the command files is provided in APPENDIX C of this manual.

It is recommended that another directory be created (D:***HP_RUN***) to compile and execute the code:

> mkdir D:***HP_RUN***
> copy D:**HPTAM***.* D:***HP_RUN***

It is now necessary to compile and link the Fortran source files of the code. On a 486 machine using the F77L Lahey Fortran compiler, for example, the compiler options are set by modifying the file **F77L3.fig**:

> fig3.exe

The recommended options are:
/n0/n2/4/n7/nA2/nB/nC/nC1/nD/nD1/nF/nH/nI/nK
/nL/O/P/Q1/nQ2/nQ3/nR/nS/nT/nV/nW/nX/Z1

The most important option is /nR, which specifies that the local variables and arrays of subprograms must not be remembered (or SAVED). Also, the /nK option prevents the generation of Weitek 1167, 3167 or 4167 code. The Lahey Fortran compiler is used to compile every source file separately:

> cd D:***HP_RUN***
> lfc **hptam.for**
> lfc **subnew.for**
> lfc **subold.for**
> lfc **poisson.for**

This compilation generates the 4 object files **hptam.obj**, **subnew.obj**, **subold.obj** and **poisson.obj**. The following command links these object files and creates the executable **hptam.exe**:

> lfl **hptam subnew subold poisson** –stack 2100000

The stack switch is used to force the maximum size of the stack segment. A stack segment of 2.1 Mb is suitable to contain all the arrays for a discretization mesh as large as 30 radial cells by 60 axial cells. After linkage, the executable file **hptam.exe** takes up 1.6 Mb of memory, so that a minimum of 1.6+2.1=3.7 Mb of free RAM is necessary to execute the code. Now, the D:***HP_RUN*** directory contains all the files necessary to execute the code.

6.2. Starting a New Calculation on MS-DOS Machine

Before a new calculation is started, the set of batch commands **clean.bat** must be executed. These commands prevent the possibility of filename collisions by renaming the existing files (the MS-DOS operating system would not allow the code to create a new file whose name already exists in the directory of interest); usually, the last (and third) character of the prefix is substituted with the character "1". Because the prefixes of filenames are limited to a maximum of three characters on MS-DOS, the filenames on MS-DOS machine have been slightly changed from what they were on the UNIX and VMS machines. The corresponding names of all the files are given in Table 2 for the three different operating systems.

> cd /tmp/***HP_RUN***
> call **clean.bat** (or simply > **clean**)

Next the input file **hp.inp** must be edited. The parameter **IFILE** must be set to **zero** when starting a new calculation. Because the inputs and outputs of HPTAM are organized to allow the user to generate a series of chained runs, there is no advantage in performing a very large number of time step iterations per run. Indeed, a reasonable value of **ITERMAX** is recommended (between 20 and 100) which gives a CPU time per run ranging between 4 and 20 mins.

Note that, unlike multi-task UNIX and VMS machines, MS-DOS machines do not allow the user to check the progression of the numerical calculation, as these machines can only perform one process at a time.

> ed **hp.inp**

Then, the first run of the code is executed:

> **hptam**

When the parameter **IFILE** is **zero**, the transient calculation is started right away if no external acceleration exists. After the specified number of time steps has been performed, the main program unit writes the time of the transient and the values of all the physical quantities at this particular time into the file **hp.sto**. However, if a gravity or acceleration field is present, the code performs internal iterations to calculate the associated non-uniform pressure field and vapor pore void fractions. After convergence of the pressure field, the time of the transient is set to zero and the values of all the physical quantities are written in the output file **hp.sto**.

After the user has created a new input file **hp.inp**, it is possible that the first run of the code will abort, due to a formatting error. In such case, the user can edit the verification file **hp.vrf** (a duplicate of the input file **hp.inp**, which contains values of the input parameters as read and understood by HPTAM), and find out which one of the formatting rules was broken. Note that additional information is provided into the output file **hp.vrf**, such as total number of numerical cells and locations of cell interfaces in both the radial and axial directions. This feature allows the user to verify that the mesh generated by the code conforms with his/her intentions.

> ed **hp.vrf**

TABLE 2. Corresponding Filenames on the Different Operating Systems.

UNIX	VMS	MS-DOS
hptam.f	hptam.for	hptam.for
subnew.f	subnew.for	subnew.for
subold.f	subold.for	subold.for
poisson.f	poisson.for	poisson.for
hp.inp	hp.inp	hp.inp
hp.ini	hp.ini	hp.ini
hp.sto	hp.sto	hp.sto
hp.vrf	hp.vrf	hp.vrf
hp.vrf1	hp.vrf1	hp.vr1
clean.com	—	clean.bat
start.com	start.com	start.bat
coop.com	coop.com	coop.bat
job*.com	job*.com	job*.bat
*.out	*.out	*.out
*.dot	*.dot	*.dot
*.time	*.time	*.tim
*.dat	*.dat	*.dat
*.fld	*.fld	*.fld
*.out1	*.out1	*.ou1
*.dot1	*.dot1	*.do1
*.time1	—	*.tm1
*.dat1	*.dat1	*.da1
*.fld1	*.fld1	*.fl1
*.tim0	*.tim0	*.tm0
job.log	job*.log	job.log
job.log1	—	job.lo1
time.log	—	time.log
time.log1	—	time.lo1

At the end of the first run, the code generates the titles, legends and necessary plot commands in the output history files ***.tim**. After the first run, the command **start.bat** must be executed, whose purpose is to prevent filename collisions, rename the file **hp.sto** into **hp.ini**, and copy every ***.tim** file (actually ***.tm1** file at that point) into the time history files ***.tm0**.

> start

6.3. Continuing a Transient Calculation on MS–DOS Machine

It is then possible to continue the transient calculation, after setting the parameter **IFILE** to **1** in the input file **hp.inp**.

> ed **hp.inp**

When the next HPTAM run is executed, the time of the transient and the values of all the physical quantities at that particular time are read from the input file **hp.ini**, and the code continues the transient calculation. After the specified number of time steps has been calculated, the new time of the transient and the values of all the physical quantities at this time are written into the output file **hp.sto**, and HPTAM generates new files ***.tim**, which contain only the data points associated with the transient period calculated. Then, the command **coop.bat** must be executed, whose purpose is to prevent filename collisions, rename the file **hp.sto** into **hp.ini**, and append every ***.tim** file at the end of the associated file ***.tm0** (using the "+" option of the MS-DOS command "copy"). To cover a large transient period, a series of chained runs can be performed by alternating code runs (> **hptam**) with **coop.bat** commands (> call **coop.bat**). The files ***.tm0** grow in size after every HPTAM run.

For convenience, job command files are included in this package (**job0.bat**, **job1.bat**, **job2.bat**, ...), which perform a series of 20 chained runs each. After setting the parameter **IFILE** to **1** in the input file **hp.inp** (the first run to start a new calculation must always be executed by hand with the command > **hptam**), the transient calculation can be continued by executing the first **job0.bat** process:

> job0

This process performs 20 code runs in succession and saves the files **hp.sto** generated by every run as **hp.001**, **hp.002**, **hp.003**, ..., to **hp.020**. It then starts the next process by executing the command **> job1**. Five jobs are chained that way automatically (**job0.bat**, **job1.bat**, ..., to **job4.bat**), for a total of 100 code runs. The code outputs generated by the WRITE(6,...) Fortran instructions are directed to the output file **job.log**, the commands and CPU time of the job are echoed in the output file **time.log**.

To interrupt a job running on the Personal Computer, the user can type the CTRL+C or CTRL+BREAK key combinations on the keyboard. To increase the number of times MS-DOS checks for these key combinations, the user can include the **break=on** command line in his/her **CONFIG.SYS** file.

After the calculation is done, the execution of the job can be checked by editing the files **job.log** and **time.log**:

> ed **job.log**
> ed **time.log**

When **IFLAG06**=1 in the input file **hp.inp**, the output files **hp.out** and **residu.out** are generated by the code. The information given in these files can be used to check the convergence of the internal iterations of the numerical procedure and evaluate the suitability of the time step for a given problem and particular period of the transient (see Section 4.2 of this manual).

> ed **hp.out**
> ed **residu.out**

While the files **hp.out** and **residu.out** contain information pertaining to the last run of the code, the files **hp.ou1** and **residu.ou1** contain information on the previous HPTAM run. These files can be visualized with the commands:

> ed **hp.ou1**
> ed **residu.ou1**

6.4. Extracting Data and Plots on MS–DOS Machine

HPTAM generates four different types of plots (Section 4.3): (a) plots ***.fld** showing axial (and radial when relevant) distribution of physical quantities in the heat pipe at the last time step calculated by the code; (b) plots ***.dat** showing the axial distribution of a given quantity at different times during the transient calculation; (c) plots ***.tm0** showing the time history of a given quantity at a given location in the heat pipe; and (d) plots of the discretized domain showing the velocity fields and/or the fluid frozen fractions in the wick (files **hpg.dot** and **hpu.dot**).

The output files ***.fld**, ***.dat** and ***.tm0** are generated in a form compatible with the EASY–PLOT software package when the parameter **IPLOTPC** is set to **1** in the input file **hp.inp**. Note that the plots **hpg.dot** and **hpu.dot** can only be visualized by the TELLAGRAF software (running on VMS machines), whereas the output files **twall.tim0** and **void.tim0** are compatible with the EASY–PLOT software (running on MS-DOS machines) only.

After execution of the main heat pipe transient processes (**job*.bat**), the user can begin to extract the data of interest. Let us assume, for example, that the user is interested into the pressure and temperature fields in the heat pipe at time 222 s of the transient. For a numerical time step **TAU**=0.1 s and a total number of time steps **ITERMAX**=50 (these parameters are specified in the input file **hp.inp**), every run of the code progresses the transient calculation by 5 s of real time. In that case, the last run of the process **job1.bat** has created the file **hp.040**, which contains the values of all the physical quantities in the heat pipe at time 200 s of the transient. Similarly, the process **job2.bat** has created the file **hp.044**, which corresponds to the time 220 s. The time of interest (222 s) can be reached by performing 20 additional iterations (with an identical time step of 0.1 s). The parameter **ITERMAX** is set to 20 in the input file **hp.inp**:

> ed **hp.inp**

The short job is performed by running the command **clean.bat** to prevent possible name collisions, copying the file **hp.044** to **hp.ini**, and running the executable **hptam**:

> **clean**
> copy **hp.044 hp.ini**
> **hptam**

The execution of the job can be checked with the commands:

> ed **job.log**
> ed **time.log**

The output files **temp*.fld** and **press*.fld** (see Section 4.3) generated by the run contain the temperature and pressure fields in the heat pipe at time 222 s of the transient. The plotting package EASY–PLOT can be used to graph these outputs.

7. RUNNING HPTAM ON VMS MACHINE

In this section, it is assumed that the DISSPLA and TELLAGRAF software packages are available on the VMS machine. The name of the user's account will be referred to as ***USERID***, and the name of the batch queue will be referred to as ***QUE_NAME***, whether it is a generic batch queue on the VMS machine (**SYS$BATCH** on DEC VAX machines, for example) or a queue on a separate cluster machine entirely dedicated to running batch processes. All the commands given in this section are functional on a DEC VAX 6000-320 machine running VAX/VMS Version 5.5–2. The user is advised to consult his/her VMS coordinator if the VMS machine is running a different operating system.

7.1. Installation of HPTAM on VMS Machine

To begin with, the user must edit and modify his/her **login.com** file:

$ vi **login.com**

First, the following lines must be added in the command section of the **login.com** file, which define the short–hand commands ***home***, ***temp*** and ***batch***:

$ ***home*** == "set def sys$login
$ ***temp*** == "set def sys$temp:[***USERID***]
$ ***batch*** :== submit/notify/noprint/que=***QUE_NAME***

The command ***home*** (or $ go ***home***) will move to the user's root directory, and the command ***temp*** (or $ go ***temp***) will move to the user's temporary space. The command ***batch*** will submit a batch process to the batch queue ***QUE_NAME***, notifying the user when the process is done, and blocking any printout to the default printer. Next, the following lines must be added in the BATCH_MODE section of the **login.com** file, to tell the batch queue that all necessary files for the batch processes must be read and written on temporary space:

$ BATCH_MODE:
$ @sys$com:mktempdir.com
$ go ***temp***

Finally, the plotting software packages must be activated in the user's account by adding the following lines in the COMMON section of the **login.com** file:

$ COMMON:
$ setup TELLAGRAF
$ setup DISSPLA

After these modifications have been made in the file **login.com**, it is necessary to activate these changes by running the command file again (this operation is performed automatically every time the user connects onto his/her account):

$ **@login.com**

After these preliminaries, the user is advised to create a new directory (***HPTAM***) on his/her main VMS account to house the code HPTAM:

$ go ***home***
$ cr/dir [***.HPTAM***]
$ sd.***HPTAM***

The File Transfer Program (FTP) can be used to transfer the files from the MS-DOS machine to the VMS directory ***HPTAM*** via the high-speed Ethernet network. After the transfer, the directory includes the four Fortran source files (**hptam.for**, **subnew.for**, **subold.for** and **poisson.for**), a standard input file (**hp.inp**), and several command files to run the code (**compile.com**, **hptam.com**, **start.com**, **coop.com**, **job*.com** files, **end.tim**, **time.com** and **tagpro.dat**). A copy of the command files is provided in APPENDIX D of this manual.

Due to the large size of the object files, the executable file of the code and the large number of output files generated by HPTAM, it is recommended to compile and execute the code on temporary disk space:

$ go ***home***
$ sd.***HPTAM***
$ copy *.* ***temp***

The user must be aware that any file in temporary space (sys$temp:[]) not used or "touched" after a certain period of time (commonly 3 days) will be automatically deleted by the operating system. It is therefore necessary to backup the wanted files onto the main disk space (when memory is available) or on a magnetic tape device, for example. When running a heat pipe calculation for several days, it is possible to reset the clock to zero by "touching" the files:

$ rename *.* *.*;1

However, this command should not be used to store on temporary space a very large number of files for a long period of time, as such practice is discouraged (sometimes sanctioned) by system operators.

It is now necessary to compile and link the Fortran source files of the code. These operations can be executed interactively on the VMS machine by typing the following commands:

$ go ***temp***
$ for **hptam.for**
$ for **subnew.for**
$ for **subold.for**
$ for **poisson.for**
$ link **hptam+subnew+subold+poisson**

or by simply executing the command file **compile.com**:

$ go ***temp***
$ **@compile.com**

It is also possible, as an alternative, to compile and link the code on the batch queue ***QUE_NAME*** by submitting the following batch process:

$ go ***temp***
$ ***batch*** **compile.com**

In that case, the compiler outputs are directed to the output file **compile.log**; this file is created in the user's root directory by the batch process, and contains additional information on the process itself, such as time of submission and CPU time.

The compilation generates the 4 object files **hptam.obj**, **subnew.obj**, **subold.obj** and **poisson.obj**. The link command creates the executable **hptam.exe**. Now, the temporary directory contains all the files necessary to execute the code.

Unlike on UNIX or MS–DOS machines, there is no danger of filename collision on VMS machines, as the version number of the new file (following the semicolon ";") is incremented by one if a namesake already exists in the directory of interest.

7.2. Starting a New Calculation on VMS Machine

Before starting a new calculation, the input file **hp.inp** must be edited and the parameter **IFILE** must be set to **zero**. Because the inputs and outputs of HPTAM are organized to allow the user to generate a series of chained runs, there is no advantage in performing a very large number of time step iterations per run. Indeed, a reasonable value of **ITERMAX** is recommended (between 20 and 100) which gives a CPU time per run ranging between 4 and 20 mins. This would allow the user to check the progression of the numerical calculation live on the VMS multi-user machine, without interrupting it. Also, it would not be necessary to rerun the complete calculation again in case a crash of the machine occurred. Instead, only the last 4 to 20 mins of computing would be lost, and the calculation could be continued from the last **hp.sto** output file generated, after the machine were rebooted.

$ vi **hp.inp**

Then, the first run of the code is executed, either interactively:

$ run **hptam** (or $ **@hptam.com**)

or as a batch process by executing the command file **hptam.com** on the queue ***QUE_NAME***:

$ ***batch*** **hptam.com**

In that case, the code outputs generated by the WRITE(6,...) Fortran instructions are directed to the output file **hptam.log**; this file is created in the user's root directory by the batch process, and contains additional information on the process itself, such as time of submission and CPU time.

When the parameter **IFILE** is **zero**, the transient calculation is started right away if no external acceleration exists. After the specified number of time steps has been performed, the main program unit writes the time of the transient and the values of all the physical quantities at this particular time into the file **hp.sto**. However, if a gravity or acceleration field is present, the code performs internal iterations to calculate the associated non-uniform pressure field and vapor pore void fractions. After convergence of the pressure field, the time of the transient is set to zero and the values of all the physical quantities are written in the output file **hp.sto**.

The following command can be used to display the user's batch processes running on the batch queue ***QUE_NAME***:

$ show que ***QUE_NAME***

The progress of the job can be checked by typing the content of the ***.log** file on the terminal:

$ go ***home***
$ type ***.log**

Note that the ***.log** file can only be typed during the execution of the process, but not edited, unless the process is done. Also, the suffix name of the ***.log** file is that of the command file submitted to the batch queue. For example, if the following command is typed:

$ ***batch process_name***.com

the corresponding ***.log** file created by the process in the user's root directory will bear the name ***process_name*.log**.

After the user has created a new input file **hp.inp**, it is possible that the first run of the code will abort, due to a formatting error. In such case, the user can edit the verification file **hp.vrf** (a duplicate of the input file **hp.inp**, which contains values of the input parameters as read and understood by HPTAM), and find out which one of the formating rules was broken. Note that additional information is provided into the output file **hp.vrf**, such as total number of numerical cells and locations of cell interfaces in both the radial and axial directions. This feature allows the user to verify that the mesh generated by the code conforms with his/her intentions.

$ go ***temp***
$ vi **hp.vrf**

At the end of the first run, the code generates the titles, legends and necessary plot commands in the output history files ***.time**. After the first run, the command **start.com** must be executed, whose purpose is to rename the file **hp.sto** into **hp.ini**, and copy every ***.time** file into the time history files ***.tim0**:

$ **@start.com**

7.3. Continuing a Transient Calculation on VMS Machine

It is then possible to continue the transient calculation, after setting the parameter **IFILE** to **1** in the input file **hp.inp**.

$ vi **hp.inp**

When the next HPTAM run is executed, the time of the transient and the values of all the physical quantities at that particular time are read from the input file **hp.ini**, and the code continues the transient calculation. After the specified number of time steps has been calculated, the new time of the transient and the values of all the physical quantities at this time are written into the output file **hp.sto**, and HPTAM generates new files ***.time**, which contain only the data points associated with the transient period calculated. Then, the command **coop.com** must be executed, whose purpose is to rename the file **hp.sto** into **hp.ini**, and append every ***.time** file at the end of the associated file ***.tim0** (using the VMS command "append"). To cover a large transient period, a series of chained runs can be performed by alternating code runs ($ run **hptam**) with **coop.com** commands ($ **@coop.com**). The files ***.tim0** grow in size after every HPTAM run.

For convenience, job command files are included in this package (**job0.com**, **job1.com**, **job2.com**, ...), which perform a series of 20 chained runs each. After setting the parameter **IFILE** to **1** in the input file **hp.inp**, the transient calculation can be continued by executing the first **job0.com** process:

$ ***batch*** **job0.com**

This process performs 20 code runs in succession and saves the files **hp.sto** generated by every run as **hp.001**, **hp.002**, **hp.003**, ..., to **hp.020**. It then starts the next process by executing the command $ ***batch*** **job1.com**. Five jobs are chained that way automatically (**job0.com**, **job1.com**, ..., to **job4.com**), for a total of 100 code runs.

During the execution of a large **job*.com** batch process, the following command can be used to display the user's batch processes and their associated process numbers:

$ show que ***QUE_NAME***

To interrupt a batch job running on the queue ***QUE_NAME***, of process number ***entry#***, the user can use the "delete" command:

$ delete/entry=***entry#***

The progress of the job can be checked by typing the content of the ***.log** file on the terminal:

$ go ***home***
$ type ***.log**

When **IFLAG06**=1 in the input file **hp.inp**, the output files **hp.out** and **residu.out** are generated by the code. The information given in these files can be used to check the convergence of the internal iterations of the numerical procedure and evaluate the suitability of the time step for a given problem and particular period of the transient (see Section 4.2 of this manual).

$ type **hp.out**
$ type **residu.out**

Similarly, while the files **hp.out** and **residu.out** contain information pertaining to the current run of the code (and cannot be edited, unfortunately), the files **hp.out1** and **residu.out1** contain information on the previous HPTAM run. These files can be visualized with the commands:

$ vi **hp.out1**
$ vi **residu.out1**

7.4. Extracting Data and Plots on VMS Machine

As the main heat pipe transient process (**job*.com**) is being executed on the batch queue, the user can begin to extract the data of interest by carrying on simultaneous calculations in a ***DISTILL*** sub-directory. This sub-directory is setup as follows:

$ go ***temp***
$ cr/dir [***.DISTILL***]
$ copy **hptam.exe** [***.DISTILL***]
$ copy **hp.inp** [***.DISTILL***]
$ copy **end.tim** [***.DISTILL***]
$ copy **time.com** [***.DISTILL***]
$ copy **tagpro.dat** [***.DISTILL***]

Let us assume, for example, that the user is interested into the pressure and temperature fields in the heat pipe at time 222 s of the transient. For a numerical time step **TAU**=0.1 s and a total number of time steps **ITERMAX**=50 (these parameters are specified in the input file **hp.inp**), every run of the code progresses the transient calculation by 5 s of real time. In that case, the last run of the process

job1.com has created the file **hp.040**, which contains the values of all the physical quantities in the heat pipe at time 200 s of the transient. After the process **job2.com** has created the file **hp.044**, which corresponds to the time 220 s, the time of interest (222 s) can be reached by performing 20 additional iterations (with an identical time step of 0.1 s). The file **hp.044** is copied to the sub-directory ***DISTILL***, and the parameter **ITERMAX** is set to 20 in the input file **hp.inp**:

$ go *temp*
$ copy **hp.044** [*.DISTILL*]
$ sd*.DISTILL*
$ vi **hp.inp**

The short run is performed interactively in the sub-directory sys$temp:[***USERID.DISTILL***] by copying the file **hp.044** to **hp.ini**, and running the executable **hptam.exe**:

$ copy **hp.044 hp.ini**
$ run **hptam**

After the run is executed, the output files **temp*.fld** and **press*.fld** (see Section 4.3) generated by the code contain the temperature and pressure fields in the heat pipe at time 222 s of the transient. The plotting package TELLAGRAF can be used to graph these outputs.

HPTAM generates four different types of plots (Section 4.3): (a) plots ***.fld** showing axial (and radial when relevant) distribution of physical quantities in the heat pipe at the last time step calculated by the code; (b) plots ***.dat** showing the axial distribution of a given quantity at different times during the transient calculation; (c) plots ***.time** showing the time history of a given quantity at a given location in the heat pipe; and (d) plots of the discretized domain showing the velocity fields and/or the fluid frozen fractions in the wick (files **hpg.dot** and **hpu.dot**).

The output files ***.fld**, ***.dat** and ***.tim0** are generated in a form compatible with the TELLAGRAF software package when the parameter **IPLOTPC** is set to **2** in the input file **hp.inp**. Note that the plots **hpg.dot** and **hpu.dot** can only be visualized by the TELLAGRAF software, whereas the output files **twall.tim0** and **void.tim0** are compatible with the EASY–PLOT software (running on MS-DOS machines) only.

Because the TELLAGRAF software requires that closing plot commands be present after the data points, the VMS command file **time.com** must be executed before the ***.tim0** history plots can be visualized on the VMS machine:

$ @time.com

The **time.com** command simply copies every ***.tim0** file into ***.tim** and appends the file **end.tim** at the end of every ***.tim** file. The file **end.tim** contains the 2 closing TELLAGRAF commands:

```
END OF DATA.
GO.
```

After the command **time.com** is executed, the history files ***.tim** can be visualized using the TELLAGRAF plotting package:

$ tella

The user must insure that the file **tagpro.dat** is present in the directory of interest. This file contains default options for the TELLAGRAF software. If no such file is available to the user, it can be created

interactively by the TELLAGRAF software itself, or the user can use the **tagpro.dat** file provided in this package, which contains the following instructions:

```
PRIMARY DEVICE IS VT240.
PRIMARY DEVICE HARDWARE CHARACTERS IS 1.
PRIMARY DEVICE DRAWING ORDER IS 0.
SECONDARY DEVICE IS POP.
PAGE LAYOUT IS LRH.
ERROR REPORTING LEVEL IS 2.
EXIT.
```

Notice that every command line for the TELLAGRAF software must end with a dot ".". More information on how to use the TELLAGRAF software for plotting graphs can be found in the TELLAGRAF manual.

APPENDIX A: SAMPLE OF THE INPUT FILE "HP.INP"

```
********** INPUT FILE OF THE 2D-HEAT PIPE TRANSIENT ANALYSIS MODEL (HPTAM) ****
* 50W, RADIATIVE  H2O STARTUP (N=100, 16x50, Lcd=60cm, Rint0<RNv, DTm=1D-8K)
*************************************************************************************

*************************************************************************************
   1       * IFILE = 1 : READ INITIAL FIELDS IN FILE HP.INI    IFILE=0 : NO     *
*************************************************************************************
   1       * IFLAG06=1 : GENERATE FILES "*.DAT", "*.FLD" FOR FIELDS PLOTTING    *
   0       * IFLAG13=1 : GENERATE COMPLETE FILES "*.OUT" FOR DEBUGGING          *
*************************************************************************************
   0.050   * NUMERICAL TIME STEP TAU (second)                                    *
    25     * ITERMAX : MAXIMUM NUMBER OF TIME STEPS                              *
  1.E-15   * EPSNORM : OVERALL CONVERGENCE ACCURACY                              *
**** INITIALIZATION PARAMETERS ************************************************
 260.00D0 * INITIAL TEMPERATURE OF HP (K)     * THE INITIAL PRESSURES ARE THE   *
   0.00   * INITIAL PORE VOID FRACTION [0,1] * PRESSURES OF EQUILIBRIUM        *
 9.954D-3 * INITIAL RADIUS OF VAPOR-SOLID INTERFACE (m)                         *
*************************************************************************************

*************************************************************************************
  5        * type of WORKING FLUID  (1=Li  2=Na  3=K  4=?  5=H2O  6=Sn  7=Ga)   *
  6        * type of WALL MATERIAL  (1=W  2=Nb 3=Zr 4=SS-304/316 6=Cu 7=Glass)  *
  6        * type of WICK MATERIAL  (1=W  2=Nb 3=Zr 4=SS-304/316 6=Cu 7=Glass)  *
  0.0      * WETTING ANGLE (DEGREE)  (GOOD W.) (0 <= ANGLE < 90) (POOR WETTING)*
  1        * type of GEOMETRY (1=CYLINDRICAL 2=SYMMETRIC SLAB 3=SLAB [BC at R=0]
*************************************************************************************
   0.00    * GRAVITY ACCELERATION   (m/s2)  [POSITIVE NUMBER, VERTICAL DOWNWARD]
   0.0     * HP INCLINATION       (DEGREE)          (EVAP->COND / HORIZON)     *
*************************************************************************************

*************************************************************************************
* DEFINITION OF THE AXIAL GEOMETRY OF THE HEAT-PIPE :                             *
*************************************************************************************
  30.0D-2 * LENGTH of the EVAPORATOR                    (m)                      *
  10.0D-2 * LENGTH of the ADIABATIC SECTION              (m)                      *
  60.0D-2 * LENGTH of the CONDENSER                     (m)                      *
*************************************************************************************
* DISCRETIZATION OF THE AXIAL GEOMETRY OF THE HEAT-PIPE :                         *
*************************************************************************************
 15        * NUMBER of CELLS in the EVAPORATOR region                            *
  5        * NUMBER of CELLS in the ADIABATIC  region                            *
 30        * NUMBER of CELLS in the CONDENSER  region                            *
*************************************************************************************

*************************************************************************************
* DEFINITION OF THE RADIAL GEOMETRY OF THE HEAT-PIPE :                            *
*     (for GEOMETRIES 1 and 2, only HALF of the VAPOR REGION is discretized)      *
*************************************************************************************
 10.00D-3 * THICKNESS of the MODELED VAPOR region   (m)                          *
 1.00D-3  * THICKNESS of the LIQUID/WICK   region   (m)                          *
 1.50D-3  * THICKNESS of the CONTAINER WALL         (m)                          *
 3.15D-3  * SPACING   of the WATER JACKET           (m) [NOT NECESSARILY USED]   *
*************************************************************************************
* DISCRETIZATION OF THE RADIAL GEOMETRY OF THE HEAT-PIPE :                        *
*************************************************************************************
  8       * NUMBER of CELLS in the VAPOR  region                                 *
  4       * NUMBER of CELLS in the LIQUID region                                 *
  4       * NUMBER of CELLS in the WALL   region                                 *
*************************************************************************************
```

```
**** POROUS WICK CHARACTERISTICS ***********************************************
  1        * TYPE of WICK (1=WIRE-SCREENED MESH   2=POWDER   3=OPEN ANNULUS)   *
   0.750   * VOLUME  POROSITY of WICK (BETWEEN 0 AND 1) [USED for TYPE=1,2 ONLY]
   0.400   * SURFACE POROSITY of WICK (BETWEEN 0 AND 1) [USED for TYPE=3   ONLY]
  100      * MESH NUMBER (per inch)                     [USED for TYPE=1   ONLY]
 -178.D-6  * WIRE DIAMETER (m)  (IF UNKNOWN, GIVE < #)  [USED for TYPE=1   ONLY]
  165.D-6  * EFFECTIVE PORE RADIUS at WICK SURFACE (m)  [USED for TYPE=2,3 ONLY]
  1.0D+09  * WICK PERMEABILITY (m2)                     [USED for TYPE=2   ONLY]
   1.00    * EVAPORATION ACCOMMODATION COEFFICIENT ( 0 <  COEFF  < 1 )          *
*******************************************************************************

*******************************************************************************
* COOLING/HEATING CONDITION AT R=0 (GEOMETRY TYPE IGEO=3 ONLY)                 *
*******************************************************************************
  1        * 1=FLUX GIVEN   2=TEMPERATURE GIVEN   3=RADIATIVE   4=CONVECTIVE   *
*******************************************************************************
* FOR EACH WALL CELL (VERTICAL CELLS AT R=0) , PLEASE SPECIFY EITHER           *
*   - the HEAT-FLUX (COOLING: Q<=0 , in W/m2)   for condition 1                *
*   - the WALL TEMPERATURE (in Kelvin)          for condition 2                *
*   - the WALL EMISSIVITY*VIEW FACTOR           for condition 3                *
*   - the jacket-fluid BULK TEMPERATURE (K)     for condition 4                *
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
*******************************************************************************

*******************************************************************************
* HEATING CONDITION AT THE EVAPORATOR :                                        *
*******************************************************************************
  1        * 1=FLUX GIVEN   2=TEMPERATURE GIVEN   3=RADIATIVE   4=CONVECTIVE   *
*******************************************************************************
* FOR EACH WALL CELL OF THE EVAPORATOR , PLEASE SPECIFY EITHER                 *
*   - the HEAT-FLUX (HEATING: Q<=0 , in W/m2)   for condition 1                *
*   - the WALL TEMPERATURE (in Kelvin)          for condition 2                *
*   - the WALL EMISSIVITY*VIEW FACTOR           for condition 3                *
*   - the jacket-fluid BULK TEMPERATURE (K)     for condition 4                *
 -2122.D0   -2122.D0   -2122.D0   -2122.D0   -2122.D0
 -2122.D0   -2122.D0   -2122.D0   -2122.D0   -2122.D0
 -2122.D0   -2122.D0   -2122.D0   -2122.D0   -2122.D0
*******************************************************************************
   1.D-1  * TlEVAP = EXPONENTIAL HEATING PERIOD (SECOND) [for condition 1 ONLY]
  0.      * TIME0  = STARTING TIME OF HEATING/COOLING (SECOND)                 *
*******************************************************************************

*******************************************************************************
* COOLING/HEATING CONDITION BETWEEN EVAPORATOR/CONDENSER (USUALLY ADIABATIC):  *
*******************************************************************************
  1        * 1=FLUX GIVEN   2=TEMPERATURE GIVEN   3=RADIATIVE   4=CONVECTIVE   *
*******************************************************************************
* FOR EACH WALL CELL BETWEEN EVAPORATOR/CONDENSER , PLEASE SPECIFY EITHER      *
*   - the HEAT-FLUX (HEATING: Q<=0 , in W/m2)   for condition 1                *
*   - the WALL TEMPERATURE (in Kelvin)          for condition 2                *
*   - the WALL EMISSIVITY*VIEW FACTOR           for condition 3                *
*   - the jacket-fluid BULK TEMPERATURE (K)     for condition 4                *
 0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0   0.D0
*******************************************************************************
```

```
*************************************************************************
* COOLING CONDITION AT THE CONDENSER :                                  *
*************************************************************************
  3       * 1=FLUX GIVEN   2=TEMPERATURE GIVEN   3=RADIATIVE   4=CONVECTIVE   *
*************************************************************************
* FOR EACH WALL CELL OF THE CONDENSER , PLEASE SPECIFY EITHER            *
*  - the HEAT-FLUX (COOLING: Q>=0 , in W/m2)              for condition 1   *
*  - the WALL TEMPERATURE (in Kelvin)                     for condition 2   *
*  - the WALL EMISSIVITY*VIEW FACTOR                      for condition 3   *
*  - the WATER JACKET charact.: Tinlet (K),FLOW RATE (kg/s) for condition 4  *
 2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0
 2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0
 2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0  2.D0
*************************************************************************
   1.D-4  * TlCOND = EXPONENTIAL COOLING PERIOD (SECOND) [for condition 1 ONLY]
*************************************************************************

*************************************************************************
*  ADDITIONAL PARAMETERS FOR HEAT TRANSFER CALCULATIONS                 *
*************************************************************************
  260.    * AVERAGE SPACE/AMBIENT TEMPERATURE (K)  (used for condition 3 only)*
* CONVECTIVE HEAT TRANSFER COEFFICIENT H (W/m2.K)  (used for condition 4 only)*
* at BOUNDARY: *  AXE Z=0  *  EVAPORATOR  *  ADIABATIC REGION  *  CONDENSER  *
               1000.D0      1000.D0        1000.D0              1800.D0
*************************************************************************


**** PHYSICAL SCHEME PARAMETERS ***************************************
    1      * VAPOR  PHASE IS ASSUMED SATURATED              (0=NO  1=YES)      *
  0        * LIQUID PHASE COMPRESSIBILITY div(Ul)   (0=NO  1=YES)              *
  0        * VAPOR  PHASE COMPRESSIBILITY div(Uv)   (0=NO  1=YES)              *
  0        * LIQUID VISCOUS DISSIPATION             (0=NO  1=YES)              *
  0        * VAPOR  VISCOUS DISSIPATION             (0=NO  1=YES)              *
    0      * FREE-MOLECULE/TRANSITION FLOW TREATMENT       (0=NO  1=YES)      *
  0.50     * ACCOMMODATION COEFFICIENT FOR FREE-MOLECULE FLOW (0 < COEFF < 1)  *
*************************************************************************
    1      * IFLAGSINK= 1 IF ONE CELL CONNECTED TO PRESSURE SINK (0 OTHERWISE) *
   12      * ISINK = RADIAL COORDINATE OF CELL CONNECTED TO PRESSURE SINK      *
    1      * JSINK =  AXIAL COORDINATE OF CELL CONNECTED TO PRESSURE SINK      *
 614.00D0  * P0SINK = SINK PRESSURE (PASCAL)                                  *
   1.D-2   * XLSINK = LENGTH OF CONNECTING TUBE (m)                           *
  1.5D-4   *  ASINK = CROSS-SECTIONAL AREA OF CONNECTING TUBE (m2)            *
  6.0D-6   *  KSINK = PERMEABILITY OF TUBE (m2) [ASINK/(8PI) for CIRCULAR PIPE]*
*************************************************************************


**** NUMERICAL SOLUTION PARAMETERS ************************************
*  GAUSS ELIMINATION PARAMETER : 1=PARTIAL PIVOTING  0=NO PIVOTING       *
  0        * IPIVOG  for MASS FLUXES (MOMENTUM BALANCE)                   *
  0        * IPIVOp  for PRESSURES                                        *
  0        * IPIVOh  for ENTHALPIES                                       *
*  GAUSS ELIMINATION PARAMETER : 1=ROW NORMALIZATION 0=NO NORMALIZATION   *
  1        * NORMRG  for MASS FLUXES (MOMENTUM BALANCE)                   *
  1        * NORMRp  for PRESSURES                                        *
  1        * NORMRh  for ENTHALPIES                                       *
*  TYPE OF MOMENTUM CONSERVATION EQUATION APPROXIMATION                   *
  1        * iSIMPLE : 0=SIMPLE     1=SIMPLEC APPROXIMATION               *
*  OPTION FOR KINETIC THEORY COUPLING : 1=YES  0=NO (MASS RATES ARE FIXED) *
  1        * iopPOISS  during SOLUTION OF POISSON  EQUATION               *
  1        * iopENTH   during SOLUTION OF ENTHALPY EQUATION               *
```

```
*****************************************************************************
   12      * IOKMAX = NUMBER OF TEMPERATURE-COUPLING INTERNAL ITERATIONS       *
   1.D-10  * CVGSIMPL = max|M'| FOR CONVERGENCE OF SIMPLEC INTERNAL ITERATIONS *
    6      * INTERMAX = MAXIMUM NUMBER OF SIMPLEC INTERNAL ITERATIONS         *
   1.D-4   * CVGenth  = max|Sh| FOR CONVERGENCE OF ENTHALPY INTERNAL ITERATIONS
   16      * ITERhMAX = MAXIMUM NUMBER OF ENTHALPY INTERNAL ITERATIONS        *
   1.D-8   * DDTmelt (KELVIN) for MUSHY REGION WIDTH (CONSTANT THROUGHOUT CAL.)*
*****************************************************************************


*****************************************************************************
* ELEMENTARY CHECK-UP OF HEAT-PIPE GEOMETRY (DO NOT FILL IN THIS PART)       *
*****************************************************************************
          * TOTAL NUMBER of CELLS in the RADIAL DIRECTION                    *
          * TOTAL NUMBER of CELLS in the  AXIAL DIRECTION                    *
*****************************************************************************
* CELLS RADIAL POSITION (START FROM FIRST VAPOR CELL) [DIMENSIONS in METER m] *
****** THE LAST POSITION IS FOR THE WATER JACKET OUTER RADIUS ****************
*****************************************************************************
* CELLS  AXIAL POSITION (START FROM EVAPORATOR)       [DIMENSIONS in METER m] *
*****************************************************************************


*****************************************************************************
* PARAMETERS FOR OUTPUT DISPLAY                                              *
*****************************************************************************
  25      * iPRINT  (STORAGE IN "#####.TIME" FILES EVERY iPRINT ITERATIONS)   *
   5      * iPRDAT  (STORAGE IN "#####.DAT " FILES EVERY iPRDAT ITERATIONS)   *
**** LIQUID PRESSURE VARIATION WITH TIME ***********************************
  12      * iLpress = i-COORDINATE OF THE POSITION CONSIDERED  (PRESSL.TIME)  *
   8      * jLpress = j-COORDINATE OF THE POSITION CONSIDERED                 *
****  VAPOR PRESSURE VARIATION WITH TIME ***********************************
   1      * iVpress = i-COORDINATE OF THE POSITION CONSIDERED  (PRESSV.TIME)  *
   1      * jVpress = j-COORDINATE OF THE POSITION CONSIDERED                 *
**** TEMPERATURE VARIATION WITH TIME ***************************************
  12      * iTEMP   = i-COORDINATE OF THE POSITION CONSIDERED   (TEMP.TIME)   *
   8      * jTEMP   = j-COORDINATE OF THE POSITION CONSIDERED                 *
**** RADIAL VELOCITY VARIATION WITH TIME ***********************************
   8      * iUr     = i-COORDINATE OF THE POSITION CONSIDERED (RADIAL.TIME)   *
   1      * jUr     = j-COORDINATE OF THE POSITION CONSIDERED                 *
****  AXIAL VELOCITY VARIATION WITH TIME ***********************************
   1      * iUz     = i-COORDINATE OF THE POSITION CONSIDERED  (AXIAL.TIME)   *
  20      * jUz     = j-COORDINATE OF THE POSITION CONSIDERED                 *
*****************************************************************************
  50.     * SCALEL (FOR TELLAGRAF FLOW FIELDS PRINTOUT, FILES "*.DOT")       *
   3.     * ARROWL (FOR TELLAGRAF FLOW FIELDS PRINTOUT, FILES "*.DOT")       *
*****************************************************************************
   2      * IPLOTPC ( 1=OUTPUT FILES COMPATIBLE WITH Easy-Plot on MS_DOS      *
********************* 2=OUTPUT FILES COMPATIBLE WITH TELLAGRAF on VMS    ) *****
```

APPENDIX B: COMMAND FILES TO RUN HPTAM ON UNIX MACHINE

```
#!/bin/csh -v

# command 'gorun' to run a job in the background on a UNIX machine.
# this command redirects code outputs to 'job.log', and
# echoes commands and print CPU time of job in 'time.log'.

mv -f time.log time.log1
mv -f  job.log  job.log1
(nohup /usr/bin/time $1 >! job.log) >&! time.log &

exit
```

```
#!/bin/csh -v

# command "showjob" to display background jobs on UNIX machine
# for user with userID "cn9gr8ai" (please CUSTOMIZE):

ps -dfa | grep cn9gr8ai
ps -f -l cn9gr8ai
exit
```

```
#!/bin/csh

# file 'clean.com', to resolve name collisions
#                between HPTAM runs on UNIX machine

mv -f hp.vrf   hp.vrf1
mv -f hp.sto   hp.ini

mv -f hjump.out    hjump.out1
mv -f liqvap.out   liqvap.out1
mv -f prop.out     prop.out1
mv -f hp.out       hp.out1
mv -f residu.out   residu.out1

mv -f hpg.dot      hpg.dot1
mv -f hpu.dot      hpu.dot1

mv -f axial.time   axial.time1
mv -f flin.time    flin.time1
mv -f fljack.time  fljack.time1
mv -f flout.time   flout.time1
mv -f llevel.time  llevel.time1
mv -f mass.time    mass.time1
mv -f mpool.time   mpool.time1
mv -f pressl.time  pressl.time1
mv -f pressv.time  pressv.time1
mv -f radial.time  radial.time1
mv -f solmass.time solmass.time1
mv -f tau.time     tau.time1
mv -f temp.time    temp.time1
mv -f thick.time   thick.time1
mv -f tpool.time   tpool.time1
mv -f twall.time   twall.time1
mv -f void.time    void.time1


mv -f evap.dat     evap.dat1
mv -f flflood.dat  flflood.dat1
mv -f pressl.dat   pressl.dat1
mv -f pressv.dat   pressv.dat1
mv -f rint.dat     rint.dat1
mv -f tint.dat     tint.dat1
mv -f void.dat     void.dat1
```

```
mv -f gzliq.fld    gzliq.fld1
mv -f grliq.fld    grliq.fld1
mv -f mach.fld     mach.fld1
mv -f press.fld    press.fld1
mv -f pressl.fld   pressl.fld1
mv -f pressv.fld   pressv.fld1
mv -f temp.fld     temp.fld1
mv -f templ.fld    templ.fld1
mv -f tempv.fld    tempv.fld1
mv -f tint.fld     tint.fld1
mv -f urvap.fld    urvap.fld1
mv -f uzvap.fld    uzvap.fld1
mv -f visdl.fld    visdl.fld1
mv -f visdv.fld    visdv.fld1

exit
```

```
#!/bin/csh -v

# file 'start.com', to be executed after first run of HPTAM on UNIX machine
#                or to clean up files in directory (name collisions)

clean.com

ATTENTION_we_REMOVE_every_old *.tim0 file

rm -f *.tim0

cp axial.time1   axial.tim0
cp flin.time1    flin.tim0
cp fljack.time1  fljack.tim0
cp flout.time1   flout.tim0
cp llevel.time1  llevel.tim0
cp mass.time1    mass.tim0
cp mpool.time1   mpool.tim0
cp pressl.time1  pressl.tim0
cp pressv.time1  pressv.tim0
cp radial.time1  radial.tim0
cp solmass.time1 solmass.tim0
cp tau.time1     tau.tim0
cp temp.time1    temp.tim0
cp thick.time1   thick.tim0
cp tpool.time1   tpool.tim0
cp twall.time1   twall.tim0
cp void.time1    void.tim0

Make_Copy_of *.tim0 into *.tim2

rm -f *.tim2

cp axial.time1   axial.tim2
cp flin.time1    flin.tim2
cp fljack.time1  fljack.tim2
cp flout.time1   flout.tim2
cp llevel.time1  llevel.tim2
cp mass.time1    mass.tim2
cp mpool.time1   mpool.tim2
cp pressl.time1  pressl.tim2
cp pressv.time1  pressv.tim2
cp radial.time1  radial.tim2
cp solmass.time1 solmass.tim2
cp tau.time1     tau.tim2
```

```
cp temp.time1    temp.tim2
cp thick.time1   thick.tim2
cp tpool.time1   tpool.tim2
cp twall.time1   twall.tim2
cp void.time1    void.tim2
```

now you modify hp.inp, RIGHT?

exit

```
#!/bin/csh

# file 'coop.com', to resolve name collisions and save transient data
#               in files '*.tim0' between HPTAM runs on UNIX machine

mv -f hp.vrf   hp.vrf1
mv -f hp.sto   hp.ini

#mv -f hjump.out    hjump.out1
#mv -f liqvap.out   liqvap.out1
#mv -f prop.out     prop.out1

mv -f hp.out       hp.out1
mv -f residu.out   residu.out1

mv -f hpg.dot      hpg.dot1
mv -f hpu.dot      hpu.dot1

mv -f axial.time   axial.time1
mv -f flin.time    flin.time1
mv -f fljack.time  fljack.time1
mv -f flout.time   flout.time1
mv -f llevel.time  llevel.time1
mv -f mass.time    mass.time1
mv -f mpool.time   mpool.time1
mv -f pressl.time  pressl.time1
mv -f pressv.time  pressv.time1
mv -f radial.time  radial.time1
mv -f solmass.time solmass.time1
mv -f tau.time     tau.time1
mv -f temp.time    temp.time1
mv -f thick.time   thick.time1
mv -f tpool.time   tpool.time1
mv -f twall.time   twall.time1
mv -f void.time    void.time1

cat axial.time1  >>axial.tim0
cat flin.time1   >>flin.tim0
cat fljack.time1 >>fljack.tim0
cat flout.time1  >>flout.tim0
cat llevel.time1 >>llevel.tim0
cat mass.time1   >>mass.tim0
cat mpool.time1  >>mpool.tim0
cat pressl.time1 >>pressl.tim0
cat pressv.time1 >>pressv.tim0
```

```
cat radial.time1  >>radial.tim0
cat solmass.time1 >>solmass.tim0
cat tau.time1     >>tau.tim0
cat temp.time1    >>temp.tim0
cat thick.time1   >>thick.tim0
cat tpool.time1   >>tpool.tim0
cat twall.time1   >>twall.tim0
cat void.time1    >>void.tim0


mv -f evap.dat     evap.dat1
mv -f flflood.dat  flflood.dat1
mv -f pressl.dat   pressl.dat1
mv -f pressv.dat   pressv.dat1
mv -f rint.dat     rint.dat1
mv -f tint.dat     tint.dat1
mv -f void.dat     void.dat1


mv -f gzliq.fld    gzliq.fld1
mv -f grliq.fld    grliq.fld1
mv -f mach.fld     mach.fld1
mv -f press.fld    press.fld1
mv -f pressl.fld   pressl.fld1
mv -f pressv.fld   pressv.fld1
mv -f temp.fld     temp.fld1
mv -f templ.fld    templ.fld1
mv -f tempv.fld    tempv.fld1
mv -f tint.fld     tint.fld1
mv -f urvap.fld    urvap.fld1
mv -f uzvap.fld    uzvap.fld1
mv -f visdl.fld    visdl.fld1
mv -f visdv.fld    visdv.fld1

exit
```

```
#!/bin/csh -v
touch *

# file 'job0.com' to start a series of chained HPTAM runs on UNIX machine

date
hptam
coop.com
cp -f hp.ini hp.001
date
hptam
coop.com
cp -f hp.ini hp.002
date
hptam
coop.com
cp -f hp.ini hp.003
date
hptam
coop.com
cp -f hp.ini hp.004
date
hptam
coop.com
cp -f hp.ini hp.005
date
hptam
coop.com
cp -f hp.ini hp.006
date
hptam
coop.com
cp -f hp.ini hp.007
date
hptam
coop.com
cp -f hp.ini hp.008
date
hptam
coop.com
cp -f hp.ini hp.009
date
hptam
coop.com
cp -f hp.ini hp.010
```

```
date
hptam
coop.com
cp -f hp.ini hp.011
date
hptam
coop.com
cp -f hp.ini hp.012
date
hptam
coop.com
cp -f hp.ini hp.013
date
hptam
coop.com
cp -f hp.ini hp.014
date
hptam
coop.com
cp -f hp.ini hp.015
date
hptam
coop.com
cp -f hp.ini hp.016
date
hptam
coop.com
cp -f hp.ini hp.017
date
hptam
coop.com
cp -f hp.ini hp.018
date
hptam
coop.com
cp -f hp.ini hp.019
date
hptam
coop.com
cp -f hp.ini hp.020
date

touch *
gorun job1.com
```

```
#!/bin/csh -v
touch *

# file 'job1.com', to continue a series of chained HPTAM runs on UNIX machine

date
hptam
coop.com
cp -f hp.ini hp.021
date
hptam
coop.com
cp -f hp.ini hp.022
date
hptam
coop.com
cp -f hp.ini hp.023
date
hptam
coop.com
cp -f hp.ini hp.024
date
hptam
coop.com
cp -f hp.ini hp.025
date
hptam
coop.com
cp -f hp.ini hp.026
date
hptam
coop.com
cp -f hp.ini hp.027
date
hptam
coop.com
cp -f hp.ini hp.028
date
hptam
coop.com
cp -f hp.ini hp.029
date
hptam
coop.com
cp -f hp.ini hp.030
```

```
date
hptam
coop.com
cp -f hp.ini hp.031
date
hptam
coop.com
cp -f hp.ini hp.032
date
hptam
coop.com
cp -f hp.ini hp.033
date
hptam
coop.com
cp -f hp.ini hp.034
date
hptam
coop.com
cp -f hp.ini hp.035
date
hptam
coop.com
cp -f hp.ini hp.036
date
hptam
coop.com
cp -f hp.ini hp.037
date
hptam
coop.com
cp -f hp.ini hp.038
date
hptam
coop.com
cp -f hp.ini hp.039
date
hptam
coop.com
cp -f hp.ini hp.040
date

touch *
gorun job2.com
```

```
#!/bin/csh -v
touch *

# file 'job6.com', to continue a series of chained HPTAM runs on UNIX machine

date
hptam
coop.com
cp -f hp.ini hp.121
date
hptam
coop.com
cp -f hp.ini hp.122
date
hptam
coop.com
cp -f hp.ini hp.123
date
hptam
coop.com
cp -f hp.ini hp.124
date
hptam
coop.com
cp -f hp.ini hp.125
date
hptam
coop.com
cp -f hp.ini hp.126
date
hptam
coop.com
cp -f hp.ini hp.127
date
hptam
coop.com
cp -f hp.ini hp.128
date
hptam
coop.com
cp -f hp.ini hp.129
date
hptam
coop.com
cp -f hp.ini hp.130
```

```
date
hptam
coop.com
cp -f hp.ini hp.131
date
hptam
coop.com
cp -f hp.ini hp.132
date
hptam
coop.com
cp -f hp.ini hp.133
date
hptam
coop.com
cp -f hp.ini hp.134
date
hptam
coop.com
cp -f hp.ini hp.135
date
hptam
coop.com
cp -f hp.ini hp.136
date
hptam
coop.com
cp -f hp.ini hp.137
date
hptam
coop.com
cp -f hp.ini hp.138
date
hptam
coop.com
cp -f hp.ini hp.139
date
hptam
coop.com
cp -f hp.ini hp.140
date

touch *

# at this point, it is possible to call "job0.com" again if one is not
```

```
# concerned with losing files "hp.001", "hp.002", "hp.003", etc ...
# gorun job0.com

exit
```

APPENDIX C: COMMAND FILES TO RUN HPTAM ON MS-DOS MACHINE

```
rem file 'clean.bat', to resolve name collisions
rem                between HPTAM runs on DOS machine
rem
echo on

copy *.log   *.lo1
del  *.log

copy hp.vrf  hp.vr1
del  hp.vrf

copy hp.sto  hp.ini
del  hp.sto

copy *.out   *.ou1
del  *.out

copy *.dot   *.do1
del  *.dot

copy *.tim   *.tm1
del  *.tim

copy *.dat   *.da1
del  *.dat

copy *.fld   *.fl1
del  *.fld
```

```
rem file 'start.bat', to be executed after first run of HPTAM on DOS machine
rem                or to clean up files in directory (name collisions)

echo on

call clean.bat

echo ATTENTION:we REMOVE every old *.tm0 file

del  *.tm0

copy  *.tm1  *.tm0
del   *.tm1

echo Make Copy of *.tm0 into *.tm2

copy  *.tm0  *.tm2

echo.
echo ----------------------------
echo now you modify hp.inp, RIGHT?
echo ----------------------------
echo.
```

```
rem file 'coop.bat', to resolve name collisions and save transient data
rem                in files '*.tm0' between HPTAM runs on DOS machine
rem
echo on

copy hp.vrf  hp.vr1
del  hp.vrf
copy hp.sto  hp.ini
del  hp.sto

copy *.out   *.ou1
del  *.out

copy *.dot   *.do1
del  *.dot

copy *.tim   *.tm1
del  *.tim

copy  *.tm0 + *.tm1

copy *.dat   *.da1
del  *.dat

copy *.fld   *.fl1
del  *.fld
```

```
rem file 'job0.bat' to start a series of chained HPTAM runs on DOS machine

echo on

echo copy hp.ini hp.000 >> time.log
copy hp.ini hp.000

time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.001 >> time.log
copy hp.ini hp.001
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.002 >> time.log
copy hp.ini hp.002
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.003 >> time.log
copy hp.ini hp.003
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.004 >> time.log
copy hp.ini hp.004
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
```

```
echo copy hp.ini hp.005 >> time.log
copy hp.ini hp.005
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.006 >> time.log
copy hp.ini hp.006
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.007 >> time.log
copy hp.ini hp.007
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.008 >> time.log
copy hp.ini hp.008
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.009 >> time.log
copy hp.ini hp.009
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.010 >> time.log
copy hp.ini hp.010
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
```

```
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.011 >> time.log
copy hp.ini hp.011
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.012 >> time.log
copy hp.ini hp.012
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.013 >> time.log
copy hp.ini hp.013
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.014 >> time.log
copy hp.ini hp.014
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.015 >> time.log
copy hp.ini hp.015
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.016 >> time.log
copy hp.ini hp.016
```

```
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.017 >> time.log
copy hp.ini hp.017
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.018 >> time.log
copy hp.ini hp.018
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.019 >> time.log
copy hp.ini hp.019
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.020 >> time.log
copy hp.ini hp.020

echo.              >> job.log
echo ---------- >> job.log
echo END of job0.bat >> job.log
echo ---------- >> job.log
echo.              >> job.log
```

job1.bat

```
rem file 'job1.bat' to continue a series of chained HPTAM runs on DOS machine

echo on

time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.021 >> time.log
copy hp.ini hp.021
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.022 >> time.log
copy hp.ini hp.022
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.023 >> time.log
copy hp.ini hp.023
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.024 >> time.log
copy hp.ini hp.024
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.025 >> time.log
copy hp.ini hp.025
time < return.inp >> job.log
```

```
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.026 >> time.log
copy hp.ini hp.026
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.027 >> time.log
copy hp.ini hp.027
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.028 >> time.log
copy hp.ini hp.028
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.029 >> time.log
copy hp.ini hp.029
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.030 >> time.log
copy hp.ini hp.030
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
```

```
echo copy hp.ini hp.031 >> time.log
copy hp.ini hp.031
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.032 >> time.log
copy hp.ini hp.032
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.033 >> time.log
copy hp.ini hp.033
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.034 >> time.log
copy hp.ini hp.034
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.035 >> time.log
copy hp.ini hp.035
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.036 >> time.log
copy hp.ini hp.036
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
```

```
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.037 >> time.log
copy hp.ini hp.037
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.038 >> time.log
copy hp.ini hp.038
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.039 >> time.log
copy hp.ini hp.039
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.040 >> time.log
copy hp.ini hp.040

echo.               >> job.log
echo ---------- >> job.log
echo END of job1.bat >> job.log
echo ---------- >> job.log
echo.               >> job.log
```

job2.bat

```
rem file 'job4.bat' to continue a series of chained HPTAM runs on DOS machine

echo on

time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.081 >> time.log
copy hp.ini hp.081
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.082 >> time.log
copy hp.ini hp.082
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.083 >> time.log
copy hp.ini hp.083
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.084 >> time.log
copy hp.ini hp.084
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.085 >> time.log
copy hp.ini hp.085
time < return.inp >> job.log
```

```
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.086 >> time.log
copy hp.ini hp.086
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.087 >> time.log
copy hp.ini hp.087
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.088 >> time.log
copy hp.ini hp.088
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.089 >> time.log
copy hp.ini hp.089
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.090 >> time.log
copy hp.ini hp.090
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
```

```
echo copy hp.ini hp.091 >> time.log
copy hp.ini hp.091
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.092 >> time.log
copy hp.ini hp.092
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.093 >> time.log
copy hp.ini hp.093
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.094 >> time.log
copy hp.ini hp.094
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.095 >> time.log
copy hp.ini hp.095
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam          >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.096 >> time.log
copy hp.ini hp.096
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
```

```
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.097 >> time.log
copy hp.ini hp.097
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.098 >> time.log
copy hp.ini hp.098
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.099 >> time.log
copy hp.ini hp.099
time < return.inp >> job.log
time < return.inp      >> time.log
echo hptam             >> time.log
hptam           >> job.log
echo call coop.bat     >> time.log
call coop.bat
echo copy hp.ini hp.100 >> time.log
copy hp.ini hp.100
time < return.inp >> job.log
time < return.inp      >> time.log

echo.               >> job.log
echo ---------- >> job.log
echo END of job4.bat >> job.log
echo ---------- >> job.log
echo.               >> job.log

rem at this point, it is possible to call "job0.bat" again
rem if one is not concerned with losing files "hp.001", "hp.002", etc...
rem job0.bat
```

APPENDIX D: COMMAND FILES TO RUN HPTAM ON VMS MACHINE

```
!## file 'COMPILE.COM' for compiling HPTAM on VMS cluster machine

$ FOR HPTAM.FOR
$ FOR SUBNEW.FOR
$ FOR SUBOLD.FOR
$ FOR POISSON.FOR
$ LINK HPTAM+SUBNEW+SUBOLD+POISSON
```

```
!## file 'HPTAM.COM' to execute first HPTAM run on VMS cluster machine

$ RUN HPTAM
```

```
!## file 'START.COM', to be executed after first run of HPTAM on VMS machine

$ RENAME *.TIME *.TIM0
$ COPY HP.STO HP.INI
$ COPY HP.INI HP.000
$
$ COPY HP.VRF  HP.VRF1
$
$ COPY *.FLD *.FLD1
$ COPY *.DAT *.DAT1
$ COPY *.DOT *.DOT1
$ COPY *.OUT *.OUT1
$
```

```
!## file 'COOP.COM', to be executed between HPTAM runs on VMS machine

$ COPY HP.STO  HP.INI
$ COPY HP.VRF  HP.VRF1
$
$ APPEND PRESSL.TIME PRESSL.TIM0
$ APPEND PRESSV.TIME PRESSV.TIM0
$ APPEND TEMP.TIME TEMP.TIM0
$ APPEND RADIAL.TIME RADIAL.TIM0
$ APPEND AXIAL.TIME AXIAL.TIM0
$ APPEND MASS.TIME MASS.TIM0
$ APPEND FLIN.TIME FLIN.TIM0
$ APPEND FLOUT.TIME FLOUT.TIM0
$ APPEND FLJACK.TIME FLJACK.TIM0
$ APPEND TAU.TIME TAU.TIM0
$ APPEND MPOOL.TIME MPOOL.TIM0
$ APPEND TPOOL.TIME TPOOL.TIM0
$ APPEND SOLMASS.TIME SOLMASS.TIM0
$ APPEND THICK.TIME THICK.TIM0
$ APPEND LLEVEL.TIME LLEVEL.TIM0
$ APPEND VOID.TIME VOID.TIM0
$ APPEND TWALL.TIME TWALL.TIM0
$
$ COPY *.FLD *.FLD1
$ COPY *.DAT *.DAT1
$ COPY *.DOT *.DOT1
$ COPY *.OUT *.OUT1
$
$ SHOW TIME
$
```

```
!## file 'JOB0.COM', to start a series of chained HPTAM runs on VMS machine

$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.001
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.002
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.003
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.004
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.005
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.006
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.007
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.008
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.009
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.010
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.011
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.012
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.013
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.014
$ RUN HPTAM
```

```
$ @COOP
$ COPY HP.INI HP.015
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.016
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.017
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.018
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.019
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.020
$
!## execute next job on VMS batch queue defined in "login.com" file:
$ BATCH JOB1.COM
```

```
!## file 'JOB1.COM', to continue a series of chained HPTAM runs on VMS machine

$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.021
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.022
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.023
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.024
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.025
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.026
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.027
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.028
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.029
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.030
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.031
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.032
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.033
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.034
$ RUN HPTAM
```

```
$ @COOP
$ COPY HP.INI HP.035
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.036
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.037
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.038
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.039
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.040
$
!## execute next job on VMS batch queue defined in "login.com" file:
$ BATCH JOB2.COM
```

```
!## file 'JOB4.COM', to continue a series of chained HPTAM runs on VMS machine

$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.081
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.082
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.083
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.084
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.085
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.086
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.087
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.088
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.089
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.090
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.091
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.092
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.093
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.094
$ RUN HPTAM
```

```
$ @COOP
$ COPY HP.INI HP.095
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.096
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.097
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.098
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.099
$ RUN HPTAM
$ @COOP
$ COPY HP.INI HP.100

!## at this point, it is possible to call "job0.com" again if one is not
!## concerned with losing files "hp.000", "hp.001", "hp.002", etc...
!## execute next job on VMS batch queue defined in "login.com" file:
!$ BATCH JOB0.COM
```

```
!## file 'TIME.COM', to be executed before visualizing transient files
!## *.TIM(0) using TELLAGRAF on VMS machine

$ PURG *.TIM
$ COPY *.TIM0 *.TIM
$
$ APPEND END.TIM  PRESSV.TIM
$ APPEND END.TIM  PRESSL.TIM
$ APPEND END.TIM   TEMP.TIM
$ APPEND END.TIM  RADIAL.TIM
$ APPEND END.TIM   AXIAL.TIM
$ APPEND END.TIM   FLIN.TIM
$ APPEND END.TIM   FLOUT.TIM
$ APPEND END.TIM  FLJACK.TIM
$ APPEND END.TIM   MASS.TIM
$ APPEND END.TIM    TAU.TIM
$ APPEND END.TIM   MPOOL.TIM
$ APPEND END.TIM   TPOOL.TIM
$ APPEND END.TIM   THICK.TIM
$ APPEND END.TIM SOLMASS.TIM
$ APPEND END.TIM  LLEVEL.TIM
$
$ COPY FLIN.TIM POWER.TIM
$ APPEND  FLOUT.TIM POWER.TIM
$ APPEND FLJACK.TIM POWER.TIM
$ COPY PRESSV.TIM PRESS.TIM
$ APPEND PRESSL.TIM PRESS.TIM
```